MULTI-ACCESS

REMOTE JOB ENTRY FOR KDF9

by

John Morvon Wilson

A Thesis submitted for the degree

of

Master of Science at Glasgow University.

ACKNOWLEDGEMENTS

In a project such as that described in this thesis almost every member of the Department contributes to a greater or lesser extent and to all members, past and present, I would like to offer my thanks for their help and cooperation. But above all two names must stand out - Prof. D.C. Gillies without whose backing the project would never have been started and without whose encouragement this thesis would never have been written; and Mr. O.I. Markovits for whose enthusiasm, friendship and tolerance I am deeply indebted.

I would also like to thank Dr. J. Rosalgrove for her careful proof reading and Dr. E.N. Maddison and Mr. D. Vigor for many hours of discussion early in the project.

C O N T E N T S

FIGURE INDEX

CHAPTER 1. BACKGROUND

Glasgow University Computing Department received an English Electric Deuce computer, the first computer delivered to a Scottish University, in November 1958 and had it replaced in April 1964 with a basic English Electric KDF9. The initial KDF9 installation was upgraded to full multiprogramming standard in March 1967 on recommendation of the Flowers Report.

For several years the work load submitted to the Department had been increasing exponentially, the load doubling roughly every eighteen months, and there was no foreseeable halt to this rising demand. In common with almost all other batch processing operating systems the turn-around-time for an average job had increased dramatically to the point where it was considered the fundamental limitation on user productivity: as early as 1964 proposals had been circulated in the Department concerning some form of cheap remote access facilities for KDF9 to reduce turn-around-time to acceptable levels and increase throughput.

The Government's acceptance of the Flowers Report in December 1965 provided an opportunity to go ahead with development of a multi-access system which would exploit

the multiprogramming facilities of the upgraded KDF9. First detailed discussions of the proposed low cost multi-access system for the University began early in September 1966, and actual hardware and software development of the Glasgow On-Line Desk (GOLD) System commenced some six weeks later with a team of four.

Individual responsibilities were assigned as follows:

Mr. W. Bowie:	provision of code conversion packages, modifications to PROMPT, implementation of the filing system.
Mr. L. Leith:	modification and segmentation of the Time Sharing Disk Director
Mr. G. Matkovits:	project leader, hardware design and development
Mr. M. Wilson:	communication and command program.

Mr. Leith and Mr. Bowie left the Department within a few months and their responsibilities were reassigned to Mr. Matkovits and myself respectively.

CHAPTER 2.1 AIMS

As the result of a proposal in Spring 1966, the Computer Board agreed to provide the KDF9 Universities with DEC PDP-8s but as the delivery date for the Glasgow machine was not until November 1967 and construction of a high speed data link was already in progress at the time, it was decided that a small hardware multiplexor would provide the best provisional interface between the KDF9 and the remote terminals of the GOLD System. This multiplexor was designed and built in the Department and was operational on four of its intended fifteen positions by Spring 1967.

The real-time processing demands of the integrated circuit multiplexor (ICM) were felt to be such that a truly interactive multi-access system based upon it could not be an economic proposition and so we turned our attention first of all to the development of a Remote Job Entry (RJE) system which would be useful in its own right as a Departmental facility but also act as a tool with which to develop a full scale multi-access system based on the PDP-8 and the KDF9. As none of the team had any great operating system experience, the RJE system, or Phase One of the GOLD system, was also intended to provide us with the experience, knowledge and ideas to carry on to the latter system, Phase Two.

2.2 PHILOSOPHY

We selected a multiprogrammed operating system as the basis of Phase One for a variety of reasons: it was clear that the existing configuration of the EXDON operating system was inadequate to support a realistic multi-access system - unsophisticated in concept, it operated on a "sausage machine" basis in an extravagant manner (the smallest of jobs still took up all of core). The early and successful development of multiprogramming techniques however provided a solution to the inefficiencies of such systems and the ideal basis for addition of multi-access systems. On KDF9, optimum core utilization (up to four programs in core simultaneously), existence of a segmented program maintenance scheme (PROMPT), provision of four multi-programmed hardware priority levels, full hardware and software program protection, existance of two ALGOL compilers (with their compact block structure) and direct user handling of most of the sixteen input/output channels were all strong arguments in favour of a multi-access system based on a multiprogrammed operating system.

In itself multiprogramming is potentially able to provide a considerable increase in throughput compared to that obtainable in multi-access systems developed in the early 1960's, which were often grafted onto standard batch processing systems.

2.3 IMPLEMENTATION

In the GOLD system the highest of the four program priority levels is assigned to the communications controller which deals with the online real time aspects of the system (running of user jobs is done by the Director and the Schedulers): this program is designed to be severely I/O limited and during the interval that access to the ICM buffers is prevented by the core lockout system, control is passed to the next highest priority level (which under GOLD would normally contain the round robin program running scheme for online jobs) or lower.

Because of the limited time and effort available it was decided to make maximum use of standard KDF9 software wherever practical and to this end we decide to utilize the Time Sharing Disk Director, KKT8ØE@@7UPU and the PROMPT program maintenance scheme, KAZ@@A71JUS4 as the basic building blocks of the GOLD system and to use a set of code conversion routines to provide the interface between the 8-bit teletype code of the ICM and the 6-bit internal codes of KDF9. These standard packages had the advantage of being fully debugged, and provided a well established program priority structure and virtually all that was needed for the creation, amendment, compilation and execution of programs in a reasonably flexible and efficient manner, relieving us of the burden of creating an operating system from scratch. However, in order to run

jobs for remote users of the system numerous modifications had to be made to these programs and the serious lack of adequate documentation proved a major delay in completion of the early phases of the work. Our decision to treat all standard KDF9 software as "black boxes" never really worked: I expended considerable time in analysing parts of the permanent program and some of the segments of PROMPT.

Wherever possible all new system facilities were written to run just as ordinary programs (under the full protection of Director and hardware) with the possibility of privileged communications with Director: this avoided the creation of a complex monolithic supervisor and, in addition, standard debugging routines could then be incorporated to provide monitoring of software operation. This was particularly true of the communications control program which always incorporated the program testing package L1. Although L1 is about 500 words in size and severely output limited, serious consideration was given to development of a version suitable for use within Director but this was not undertaken due to lack of time.

Program mode operation of the communications controller also had the great advantage that it could be run as a normal user job with a simulated IOM during development under the standard operating system, at no inconvenience to the operators.

In design and development of the controller, I attempted to retain a modular form of construction throughout, splitting it up into a number of functionally independent subroutines with a common workspace and constants storage area. Although contributing to more rapid debugging and easier extension and modification, the prime purpose of this exercise was to facilitate segmentation of the controller at a later date in its development: modularity also meant that very often additional functions in the controller could be provided quite readily by linking chains of suitable subroutine calls with very little extra coding.

The Director and component programs of the GOLD system required intimate, secure intercommunication and this was achieved by means of a system of privileged OUTs. These were assigned negative values to distinguish them from standard OUTs and as a protection against the possibility that English Electric might issue new OUTs with the unallocated positive values. The possibility of program intercommunication by suitable settings of base address and number-of-location registers to give an element of core sharing between adjacent programs was dismissed early on as being complex to administer and, in the case of program-Director core sharing, dangerous. OUT's were provided to fetch and store information in the body of

of Director, to identify system programs (see OUT-7, OUT-9), initiate job queues etc. (See Appendix III).

CHAPTER 3 PHASE ONE : Remote Job Entry System.

Phase One consisted of a suite of programs comprising

(a) the segmented multiprogramming disk director and the segment loader

(b) the pseudo-off-line loader used to spool jobs to disk, its overlay and the job organiser

(c) the communications controller and loader

(d) teletype PROMPT and the editor

and one special purpose piece of hardware - the integrated circuit multiplexor, ICM. Mr. Matkovits was responsible for (a) and (b) and the ICM: I was responsible for (c) and (d).

In Phase One, two levels of multiprogramming were permanently assigned to component programs of the CDLD system under control of the Director: stack P (normally priority 0) was occupied by the communications control program, CommCon; stack S (normally priority 3) by the pseudo-offline loader. Director provided four job queues, one per program stack, each acting as an input well and all of them serviced by the schedule JOBORG---UPU (See Appendix III) CommCon and the POL loader each used the privileged OUT-4 to insert job requests into their

respective disk based job queues: each request consisted of a 40 word sector of information known as the Job Heading Sector, (JHS). When any priority level became free (normally only in stacks Q and R) Director loaded the job organising program, JOBORGØ13UPU, which scanned these queues and selected the next appropriate job to be run. JOBORG informed Director of its choice by an OUT-6, loading job details into V17-25 of P120 in Director, and ended. Director then loaded and ran the selected job, output being sent to user specified devices. A GOLD system user could obtain immediate return of output to his terminal by means of the standard OUT+8 facility but using a stream number of -1 or -2.

A storage map of the operating system is shown in Figure 1 .

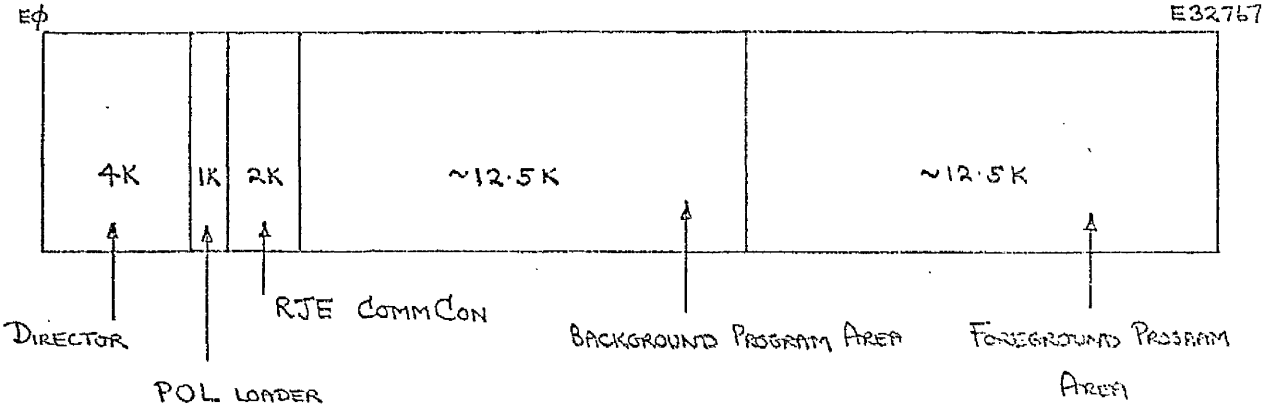


Fig 1 : allocation of main store in the GOLD system.

My work on the communications controller went through three distinct stages of evolution

(i) a disk-based version using magnetic tape for intermediate storage of input data

(ii) an all-tape version supporting one remote terminal used at a demonstration when the disk file became totally unserviceable

(iii) the final all-disk version

This thesis is primarily concerned with the third of the above and details the achievements and shortcomings of the Remote Job Entry phase of the GOLD system as it was when abandoned in November 1967. For completeness details of Mr. Matkovits' work are included in Appendices II, III, and summaries of (i) and (ii) above appear in Chapter 6 and Chapter 7 respectively.

CHAPTER 4.1 GENERAL DESCRIPTION OF COMMUNICATIONS CONTROLLER.

The GOLD system communications controller, referred to hereafter as CommCon, was designed to control all real-time character input/output for upto fifteen terminals connected via a common data bus to the IOM. It also had to provide buffer organization, command handling and execution, management of temporary file storage in the input and output wells, and the assembly and queuing of jobs (JHS) specifications in its input well. All this had to be done in 2K words of core and with minimal CPU demands.

Although CommCon was designed to handle fifteen terminals, initial effort was directed at implementation with coding and constants for only four. This was quite adequate in the initial stages because hardware was not available for more than four terminals when development testing got under way. Full provision in terms of coding and buffer space for all fifteen was made, however, and the only required changes to CommCon to accommodate the additional terminals as they were connected were minor alterations in the output routines and the CommCon constants table, P899.

4.2 OPERATION OF COMMCON

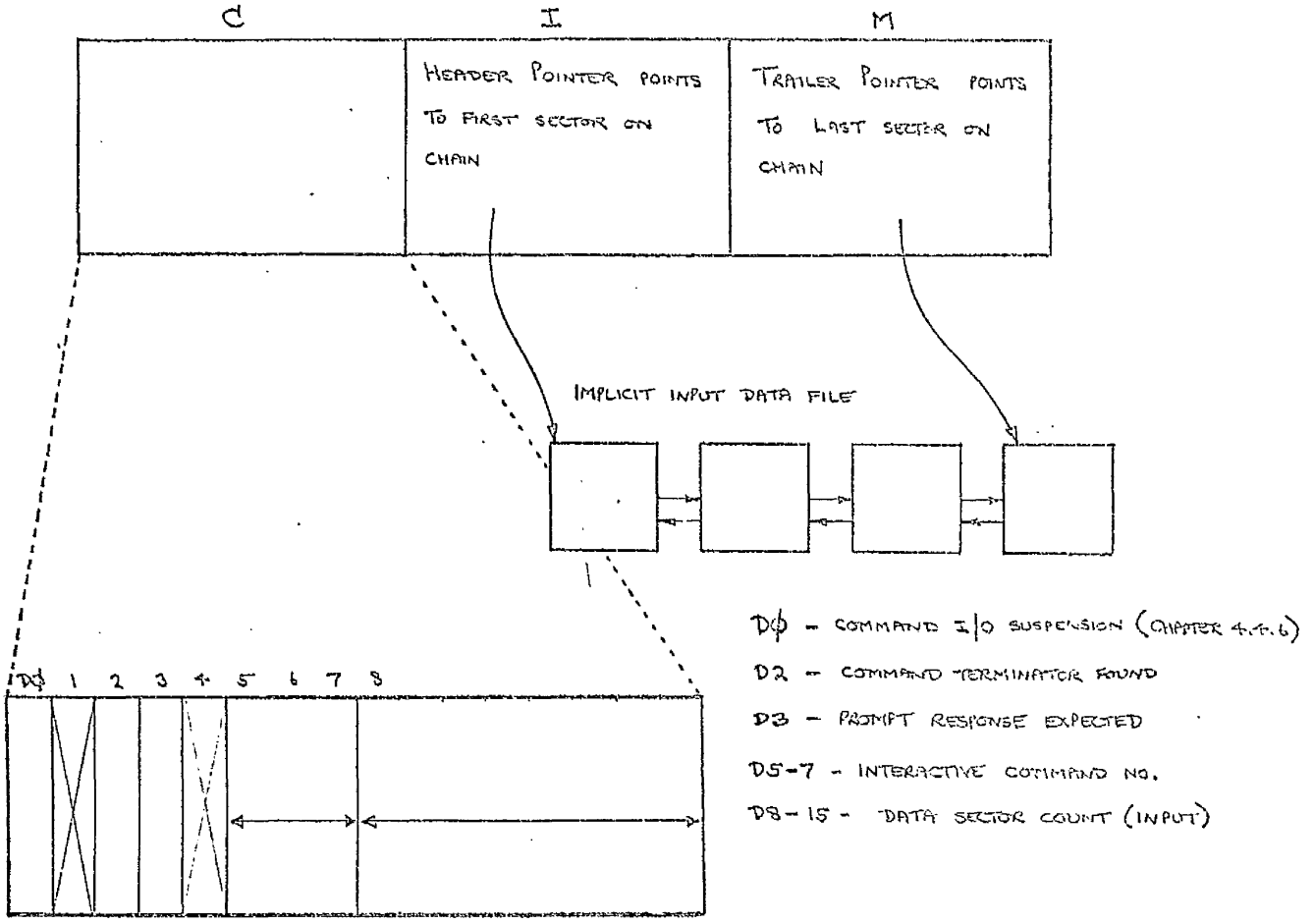
The input facilities of CommCon consisted of a pair of double buffered ICR transfer areas which, in turn, fed a set of software first-in-first-out stores; the passage of certain characters from the former to the latter could set certain marker bits in a pair of control words, one pair per terminal. The two words of each pair were known as Desk Flag Word 1 (DFW 1) and Desk Flag Word 2 (DFW 2), and contained the information shown in Fig. 2. As well as character markers etc., DFW1 and DFW2 controlled the creation and chaining of the temporary data files in the input well which held the character string typed by each user at his terminal.

All temporary data files held under CommCon were stored as a doubly linked list structure for two reasons

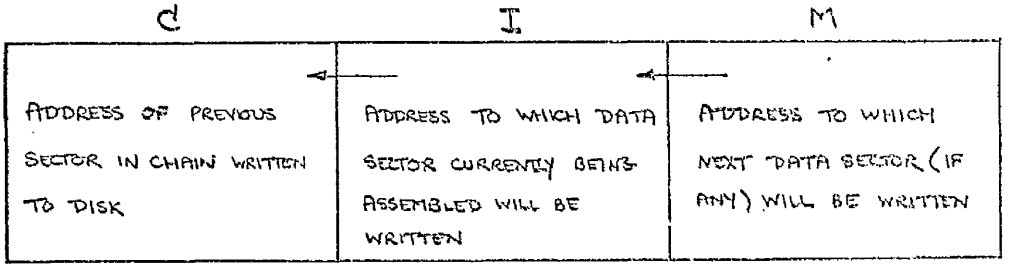
(i) the forward link (older to newer, lexicographically) allowed a simple form of dynamic allocation of disk space

(ii) the backward link (newer to older, lexicographically) allowed the operation of a backward scanning editing program for text amendment and manipulation.

From the FIFO for each terminal input data



DESK FLAG WORD 1



AS EACH DATA SECTOR BUFFER IS FILLED AND WRITTEN TO DISK, DFW2 I-PART SUPPLIES THE INTENDED DESTINATION. DFW2 IS THEN SHL+16' AND A NEW M-PART GENERATED -- THE OLD M-PART BECOMING THE NEW I-PART AND INTENDED DESTINATION OF THE NEW DATA SECTOR BEING ASSEMBLED.

DESK FLAG WORD 2

FIG. 2. DFW1, DFW2.

was moved 6 eight bit characters at a time (one KMF9 word) to that terminal's 40 word disk sector buffer under the control of its sector packing control word (SPCW) held in P898 and thence into temporary chained disk data files under control of DFM2. The temporary data files were not explicitly known to a user and constituted an input roll or job queue.

Output facilities were under control of a pair of output control words, one pair per terminal, known as the Output Control Word 1 (OCW1) and Output Control Word 2 (OCW2). They hold the information shown in Fig. 9. Output was buffered via the same 40 word core area as for input buffering for each terminal. This was possible because a terminal was deemed to be in either of two distinct states, sending or receiving, and its current state (and therefore its use of the 40 word buffer assigned to it) was recorded in V54P899. Control of terminal-directed output from a job still active in the KMF9 was further effected by an interprogram communication block (IPCB) held in P896. Output data was read from the chained files in the disk well, under control of certain counters in the IPCB and OCW2, into the output buffer and moved from there, in groups of about 10 characters at a time into the IOM output buffers under control of OCW1.

The next section describes in detail the philosophy, organization and operation of the communications controller.

4.3 MAIN CONTROL

The unfailing servicing of I/O transfers on the multiplexor is the responsibility solely of CommCon and this requirement had a great influence on its design. Being a real-time program, queues of processing requests must be allowed to accumulate in CommCon and these are maintained as sets of markers to be detailed later.

CommCon double buffers the ICM on alternate pairs of a set of four buffers, (each buffer in a separate block of 32 words for reasons of lockout) and provided it finds no outstanding processing requests in its queues that may now be completed (e.g. output waiting, commands to be decoded or completed) attempts to fetch the first word from the current ICM input buffer. As the ICM transfer is almost certainly still in progress after any such processing is complete, a lock-out violation occurs, Director is entered and control passes to the next free priority level. If there are outstanding requests they are completed by CommCon one by one, a check being made after each request is completed that the ICM transfers are still running. The check is made in P900 by testing lockout (TLOQq) on the current ICM input buffer specified in V35P899. This system of processing each request and inspecting the state of the ICM transfer before processing the next is akin to allowing an interrupt, but in a

software sense. As implemented, following a software interrupt queue scanning always resumes with the input queues and for the lowest addressed terminal first. Terminals with numerically higher addresses may occasionally experience short pauses during output activity if large processing request queues have built up which cannot be worked off before conditions for a software interrupt arise. As each processing request is deliberately designed to be of short duration or capable of suspension this drawback should not be intolerable: it was never detected in the operational system.

4.4 SUB CONTROL

The functions of CommCon fall into three main areas

(i) character input and the input data wall

(ii) character output and the output data wall

(iii) command handling and execution

Following a brief description of the initialization of CommCon, each of these topics will be considered in detail.

4.4.0.1.

Communications Controller Loader.

This was an unprivileged B-type program called into program stack P (priority Ø) in response to a TINT T: its task was to set up various constants for CommCon, clear core and working registers, claim and initiallize the ICM, claim disk working space (a total of two physical disk platters or 12,288 sectors), load a set of command descriptions and system messages to disk and assemble the data (as given in E7 of the program B-block) in teletype characters. The loader then ended via OUT+1 and called in CommCon itself.

4.4.0.2.

Communications Controller -- CommCon.

Entry into CommCon via OUT 1 from CCL found the current date in teletype characters in N1, N2: the ICM output channel number in N3 and input channel number in N4. The initial phase of CommCon used these data to complete its constants table P899 (roughly the analog of Director's P104); then OUT-9 handed over to Director the base address of the Inter Program Communication Block, IPCB, holding the GOLD Output Availability Words (GOAW) and the Director Output Availability Words (DOAW).

Immediately prior to initiation of the ICM transfers, P803 was entered to check the result of any outstanding TINT B for this priority. Unless deactivation of the teletype network had been selected by TINT B1, EXIT 2 from P803 lead back to the ICM I/O instructions PWQ15 and PRQ15. (Hardware considerations dictated their execution in this order.)

4.4.1 INPUT ORGANIZATION

To illustrate the input organization of CommCon it is best to follow the path of control through the program immediately control is returned to it by Director upon completion of the ICM transfers and ending of the LOV condition.

The first word is fetched from the ICM input buffer for the transfer that has just terminated and the most significant twelve bits extracted: each twelve bit field has the structure drawn in Fig. 3 .

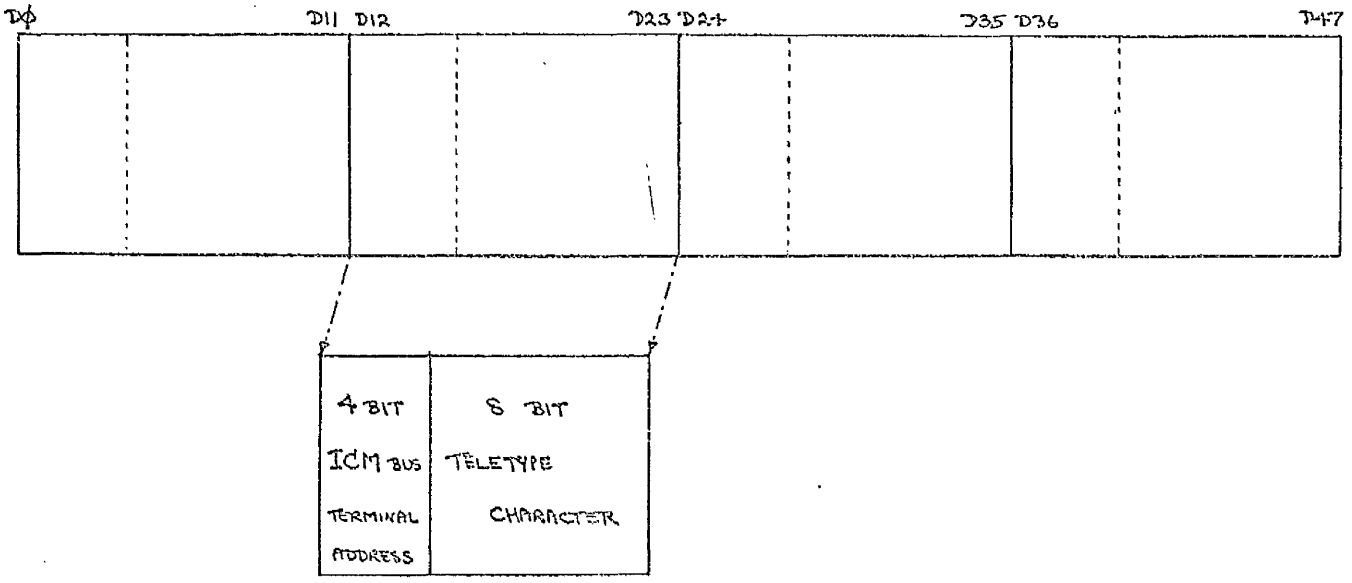


Fig. 3 Substructure of 12 bit ICM field.

If the 8 bit character field is found to contain the pattern corresponding to the erase character (D40-D47 all ones) then this field is ignored and the inspection repeated with the next 12 bit field until the word is emptied, when the next buffer word is fetched. If the field does not contain the erase character, 377_8, then it is checked for the presence of the character "@" (300_8) or the character "½" (243_8): if neither the character is passed onto the software FIFO routines. However, if either of these patterns is detected appropriate action to be detailed later takes place.

When this cycle is fully complete and all buffer words have been dealt with (the number of words is given in P899, the constants table), ComeCon deals with the processing request queues until either the queues are exhausted, or all entries are suspended (e.g. awaiting completion of I/O transfers on their behalf), or a software interrupt condition is dtected: the cycle is then repeated.

The system of software first-in-first-out (FIFO) stores was developed as the solution to a variety of problems concerned with character input to the GOLD system: these problems were:

(a) there should be no restriction upon the rates at which a terminal user entered character strings

(b) there should be a simple method to retain the character string representing a command in core until the string is correctly terminated, without either the inefficiency of handling complete lines of text (poor utilization of storage) or the problem of double buffering (command names may straddle buffer boundaries)

(c) while it is easy to recognise a command warning character (one which precedes the command name as the "£" in "£NEW.@" in the GOLD command format) and in some sense alert the system for the rest of the name,

evaluation to proceed upon the receipt of a suitable terminating character, this involves the storage of these strings in a manner differing from normal - an unnecessary complication if one structure could fulfill both needs.

(d) there should be some simple mechanism to cope with small quantities of information entered by the user in response to system prompts or queries.

(e) some storage scheme is necessary which permits the user to delete the last few characters or character he typed. The system devised is a compromise between line deletion and single character deletion.

The basic operation of the FIFO store system was as follows:-

From P800, when a character from the ICM buffer was handed over to the FIFO routines in P810, its associated terminal address was used to access three words from a table of FIFO stores (See Fig. 4). These three words made up the FIFO store for that terminal and were known as the new word, the old word and the dump word. The oldest (most significant) 8 bit character in the old word was shifted into the newest (least significant) character position in the dump word and the dump word returned to its place in the FIFO store table, provided the dump word was not greater than or equal to zero. This was to allow a short cut method

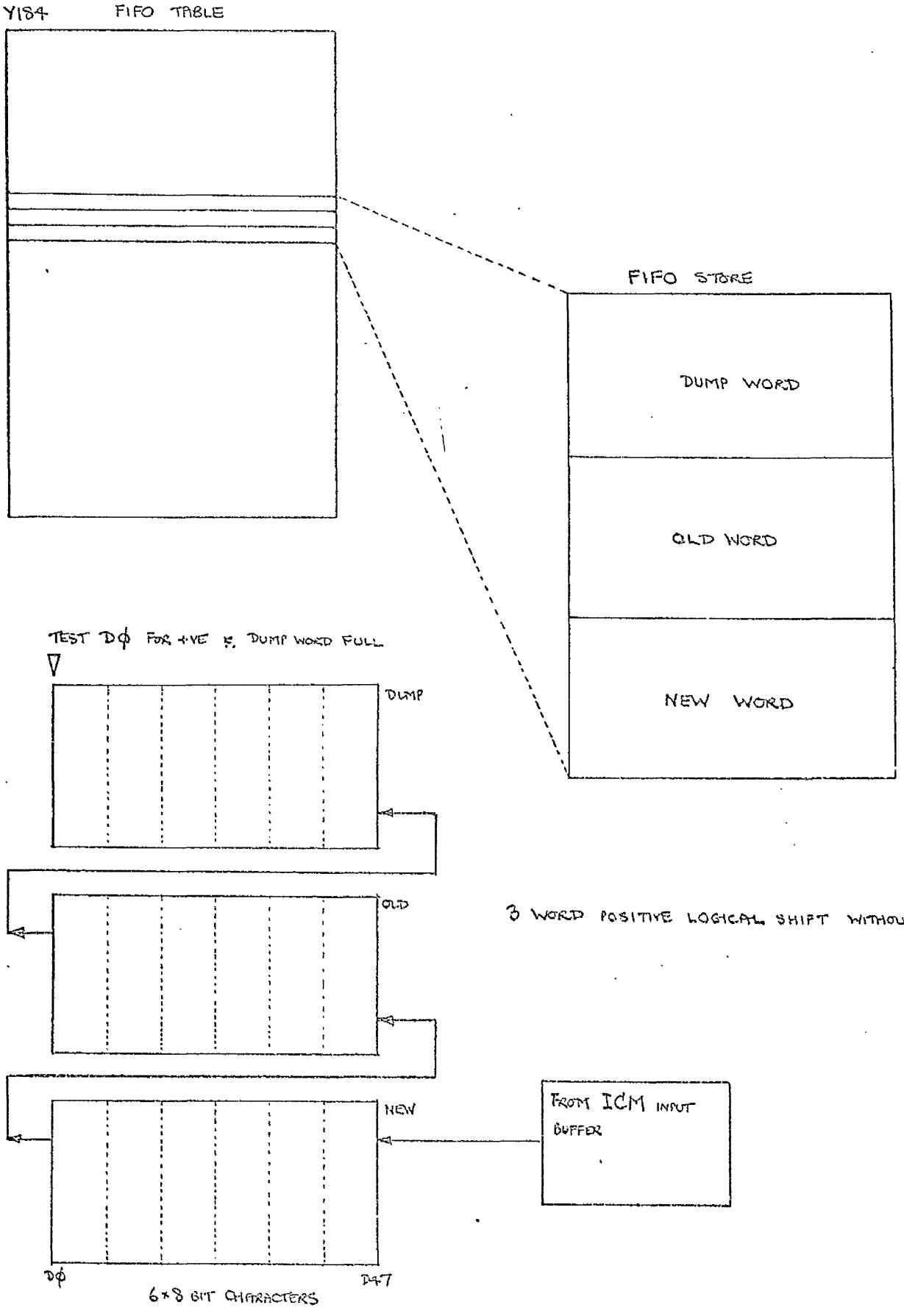


FIG. 4 STRUCTURE OF FIRST-IN-FIRST-OUT STORES.

of counting characters entering the dump word: instead of an explicit count being maintained, the binary pattern corresponding to -256_{10} was originally held in the dump word. This pattern has the octal value 7777 7777 7777 7400 and consisted of five 377_8 and one 000_8 characters which are dummies in the GOLD system. Up to five successive insertions of 8 bit character into the least significant end of the dump word could be made before its sign bit DO become zero as the 000_8 character reached the most significant end. At this point one more character filled the dump word and it was copied into the 40 word data buffer under control of the SPCWs in P898: the dump word position of the appropriate FIFO was then reset to -256_{10} and "counting" resumed. By shifting, the oldest character in the new word was moved into the newest character position in the old word and the old word restored in the table, and finally the new character was moved into the newest character position in the new word and it too restored in the table.

It can be seen that no character can get into the data buffer before it is at least the twelfth character in the FIFO store (i.e. the twelfth before the one just typed): at all times at least the last twelve characters entered are available in core irrespective of the users

typing speed or any backing store transfers: provided a command name is less than twelve characters in length it may always be found in core - in fact keeping the name even shorter means that there is some leeway in the event of a software interrupt occuring when the user who issued the command keeps on typing very fast. In such circumstances he would have to type at least six characters within one or two seconds to have any chance of losing the oldest characters of a six character command name into the dump word and the dump word would itself have to be filled before they were lost into the backing store buffer; in this case the command could not be recognised and the user notified - the leading symbols of the command name would be in the data buffer and could cause corruption of the preceding data string but this was the users responsibility.

Character strings which are parts of a command name and strings which are to be modified by means of the "$\frac{1}{2}$" character correction facility (See Chapter 4.4.1.1) are stored indistinguishably in one structure: by suitable clearing of the dump and old FIFO words and setting of the new word to -256_{10} up to eighteen character positions became available for short user responses to system prompts: a simple but effective error correction facility (the "$\frac{1}{2}$" facility) becomes available at very little cost; and finally the FIFO store acts as a simple command name buffer.

4.4.1.1.

If the "@" or ";" symbol is detected in the input buffer the following actions take place

(i) the "@" symbol: detection of this symbol has two interpretations

(a) as the terminator to a command in the FIFO store

(b) as the terminator to a response to a system prompt in the FIFO store.

In (a) detection of the command terminator alone allowed a considerable simplification in the logic of the command decoder for there was no need to provide two separate destinations for character strings, e.g. one for strings following a "£" symbol (and assumed part of a command), and one for all others: rather all characters were fed directly into the FIFO. This also had the advantage of reducing the number of different symbols which had to be detected during input processing. Upon receipt of the "@" symbol D2 of DR#1 (See 4.4.5.) for the relevant terminal was set to one to indicate that this terminal had issued a command that required decoding and a one word marker (V2#099) was set all ones to indicate that there was at least one entry in the command decode queue. The ";" symbol was then stored into the FIFO and processing continued as before.

No attempt was made at this stage to check for the initial "&" symbol or for a valid command name and this made possible the second interpretation, (b).

In (b), as above, D2 of DFW1 for the terminal and V2P899 were set. Interpretation as a prompt terminator was not made until the command decoding routines inspected D5 of DFW1 (See 4.4.5.).

(ii) the"½" symbol: detection of this symbol had two interpretations

(a) if D3 of DFW1 of the relevant terminal was not set then the symbol was interpreted as a command to delete the new word from the appropriate FIFO store, i.e. the user had made a typing error which he wished to delete: as the presence in the FIFO store of non-printing characters (CR, LF, Space, etc) could make user determination of what had been deleted uncertain, the dump and old words were typed back to the user who appended the correct characters and carried on.

(b) if D3 of DFW1 of the particular terminal was set then the command decode routines were expecting the reply to a prompt and the "½" symbol was interpreted as meaning the user wished to retype an incorrect response to a prompt. The FIFO was reset and the prompts

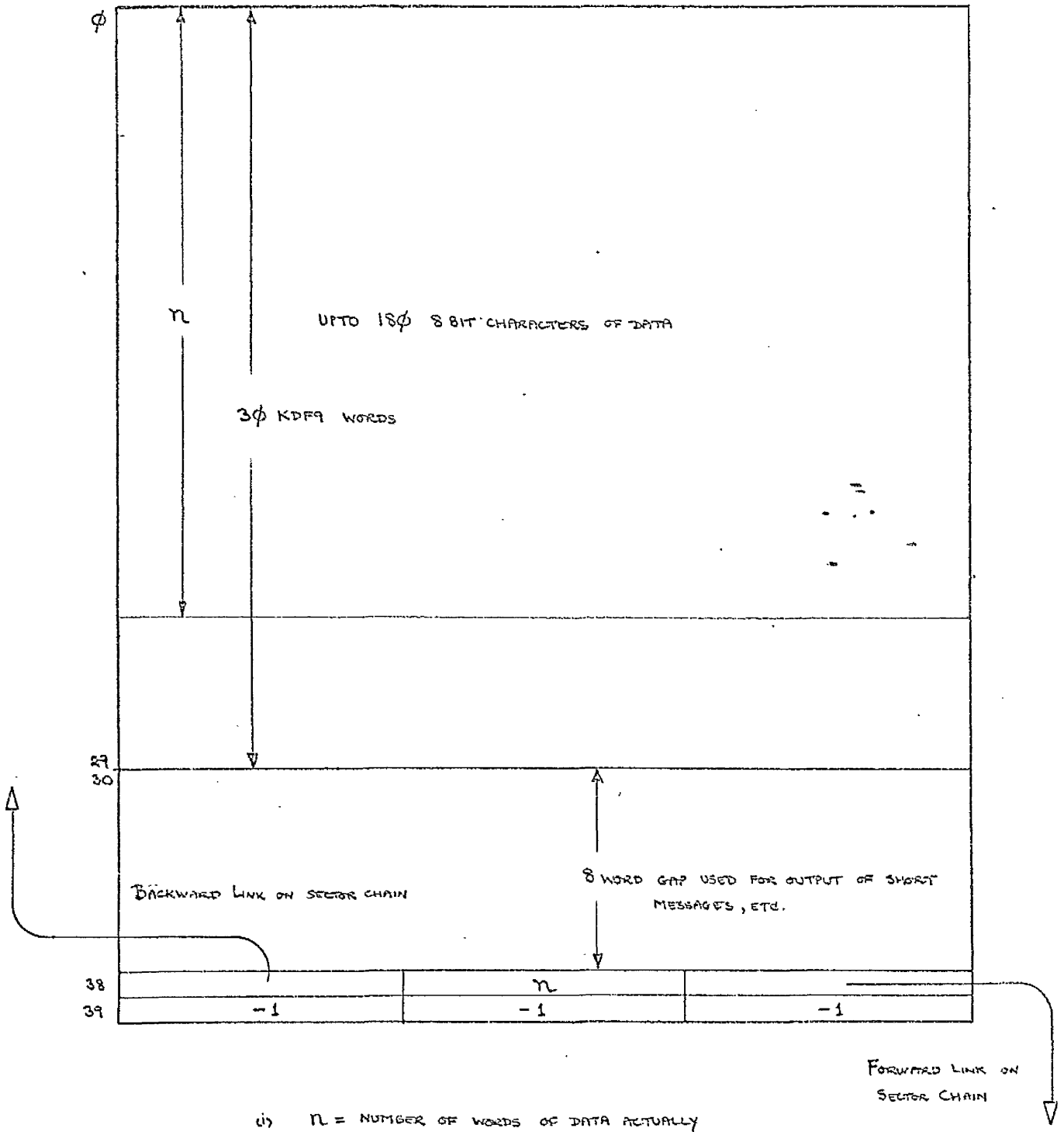
re-issued. (See page 32.).

4.4.2 THE INPUT WELL -- Temporary storage of Input Character Strings.

When the specifications of the KDF9 disk file were first made available to us we noted what appeared an obvious bottleneck in the system: although the transfer rate was quite fast (upto 96Kch. per second), the seek time could be very long (up to 350 msec.): since we had originally hoped to keep Director segments, user files, Comdon segments, binary programs, the round-robin program running scheme and its contents and the PROMPT system all on the disk it seemed obvious that excessive delays would develop in the Director disk access queues and choke the operating system. We decided that the disk channel load could be lightened by

(a) running the round-robin between two swinging tapes and (b) temporarily storing character strings typed by users on yet another tape, fetching them to disk only when required: this was called the Input Well Magnetic Tape (IWMT) System.

Implementation of (a) has never begun because straight away it ruled out any form of effective scheduling and realistic user interaction with a running job - this was unacceptable.



(i) n = NUMBER OF WORDS OF DATA ACTUALLY OCCUPYING SECTOR

(ii) E39 IS SPARE

(iii) ZERO CHAIN ADDRESS FORWARDS OR BACKWARDS MEANS END-OF-CHAIN

FIG. 5. LAYOUT OF 40 WORD DISK SECTOR AS PART OF A CHAINED DATA FILE.

Implementation of (b) was begun and continued for nearly ten months before being abandoned in October 1967 as slow, cumbersome and illconceived. (See Chapter 6 IWMF System). At this point I wrote a simple straight forward pure-disk system that was working within two weeks. In this pure-disk system temporary data storage consisted of chains of 40 word disk sectors allocated 1 block (16 sectors) at a time: each sector was filled in turn from a 40 word sector buffer, one per terminal, of which the first thirty words hold the characters making up the strings typed at that terminal and the final two words hold sector chaining and structure data as shown in Figure 5.

A running sequence of forward link address, current sector address and backward sector link address was kept as a 3-store format word in each DF%2 in the Desk Flag Word Table in AY168 onwards. When a user issued a "NEW" command the two DF% entries corresponding to his terminal in the DF% table were reset, a new disk data block was allocated to his terminal by the Allocated Space List routines (P801) as a temporary input data file and DF%2 set as shown in Fig. 6.

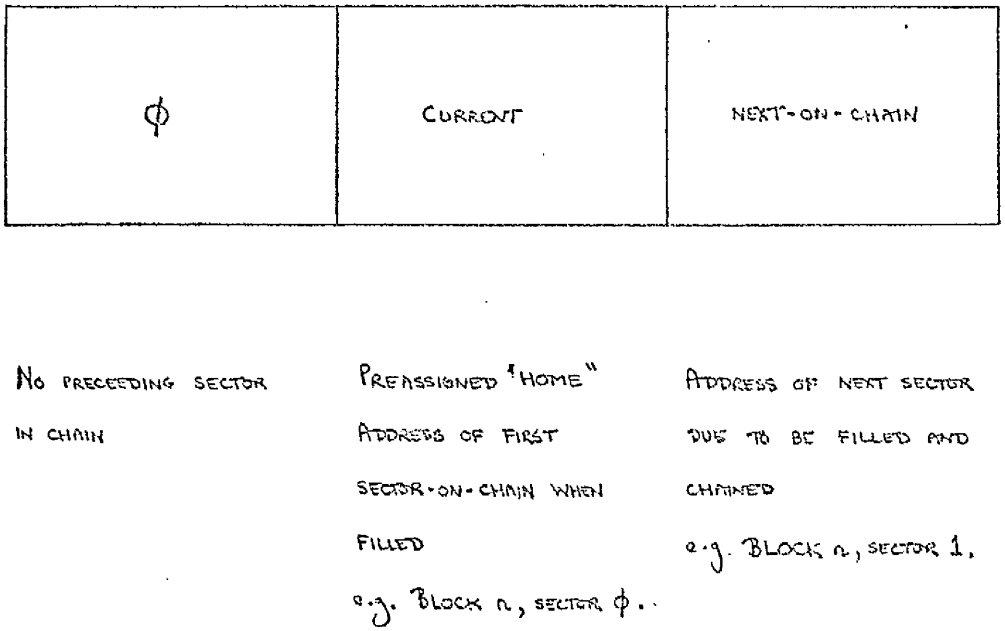


Fig. 6.

Layout of DFW2 following "NEW" command.

As a user typed, characters were packed 6 at a time (one word) into consecutive sectors of the block: if sixteen sectors were filled and written to disk thereby exhausting space in the block the ASL was referenced and a further block chained on and assigned to the terminal.

Any command issued by the user which required the preceeding character string terminated the chaining with an empty forward link and set the first-address-on-chain and the last-address-on-chain in the I and M parts of DFW1 (See Figure 2) to provide file access to the code conversion package and the backward scanning editor respectively. Input data now existed as an implicit file

in the disk input well where it remained until read by a user job.

There follow short descriptions of a number of routines used in input control; the reader is referred to Appendix 1 for flow charts and coding.

P800: the main input subroutine is described above.

P900: given the addresses of the current ICM input buffer check that lockout is still on: if not then EXIT 1 to give a software interrupt, else EXIT 2 to continue processing.

P905: reset the inactive ICM output buffer to address-erase- fields as shown in Fig. 3 , four per word in preparation for AND-ing of output characters and setting up of next output transfer (P808).

P806: given address of a terminal in M1, sets the base address of its disk sector data buffer in M9.

P801: control the usage of space on the two physical disk platters allocated to CommCon: this space includes temporary file storage, job heading sector queues and CommCon permanent data files. Space was represented by an Allocated Space List (ASL) in V15 to V30 consisting of 8 words per disk, each bit representing one 640 word

(16 x 40 word sectors) disk block in terms of its logical disk address by the relative position of that bit in the ASL. A further list in V0 to V14 held for each terminal a word which contained the next-sector-on-chain address. If an allocated block was filled, or when a user issued a "NEW" command, the ASL was searched for the first empty block and its address inserted into the next-sector-on-chain address word in V0-V14.

P802: packed dump word from a FIFO into the data buffer under control of the sector pack control words in P898 (Fig. 7).

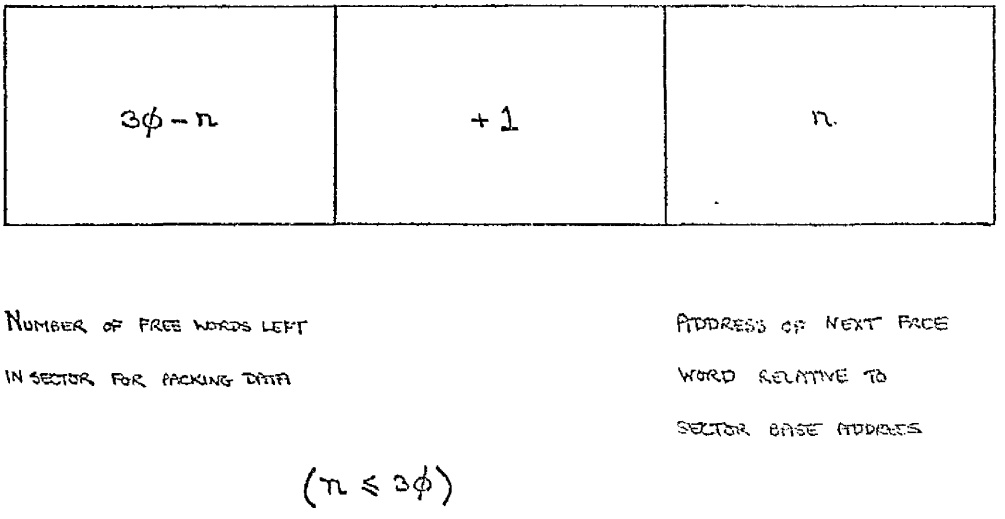


Fig. 7. SPCW - Sector pack control word - in P898

A side-entry at reference 1, reset the SPCW, set up the sector chaining information in the data buffer, updated the sector count in DFW1, used OUT 41 to copy the buffer to the current sector address on disk as specified in DFW2, and updated DFW2 chaining data.

P803: this routine allowed a TINT B to control the use of the ICM.

TINT B1 - deallocated the ICM channels, typed a warning message to the operators and went into a loop awaiting a further TINT B. The ICM could now be allocated to another program for use in burst mode operation while the RJE system was suspended.

TINT B2 - reclaimed a type 65 and type 66 device (the ICM channels), typed a message to the operators and restarted the RJE system.

TINT B3 - permitted the operator to query the status of the GOLD system, i.e. RJE active or suspended.

P807: set D2 of DFW2 for a terminal issuing the command terminator "@"; simultaneously V2P899 is set all ones to force entry to the command decoder, P821, and a one bit marker in V34P899 is changed

from 0 to 1 indicating that the terminal in question has gone from a conceptual transmit-to-KDF9 mode to a conceptual receive-from-KDF9 mode and awaits output from ComCon.

P810: stores a character into the appropriate FIFO store (see above)

P811: checks B3 of DFH1 to see if a reply to a system prompt is due: if so, it further checks that a prompt cycle reset is permitted and if so sets to zero B11 of the job heading sector being assembled for this terminal to restart the cycle of system prompts. (This is the second interpretation of the "$" facility). B11 in the JHS buffer acts as a switch to enter coding to process each response; setting it to zero merely causes all the prompts to be issued again, but a marker is provided in B22 which, if non-zero, does not allow B11 to be reset in this way. B22 is set non-zero whenever conditions arise which would make re-issuance of prompts upset the internal logic of ComCon (e.g. "CALL" command final prompt "PRINT added Y/N;") See Chapter 4.4.6.1. If a reply to a prompt is not expected, the output status bit in V54P099 for this terminal is changed to "receive", the new FIFO word reset, the old and dump words copied and set out

as output in the data buffer work space (Fig 5) and OCW1 set up to print back these copies of the FIFO old and dump words to the users terminal. Note that a terminal uses one 40 word data buffer for both input and output since it cannot both send and receive bulk data simultaneously. It can send a character when receiving ("@" acts as a quit-printing command) or receive a few character when sending ("½" returning the dump and old words of the FIFO). The unused eight words in the buffer (see Fig 5) are used for short output messages, etc.

P898: contains the sector pack control words, SPCW. Initially of the format Q30/1/0, these words define the amount of space available in the terminal data buffer: they are used by P802 (See Fig. 8)

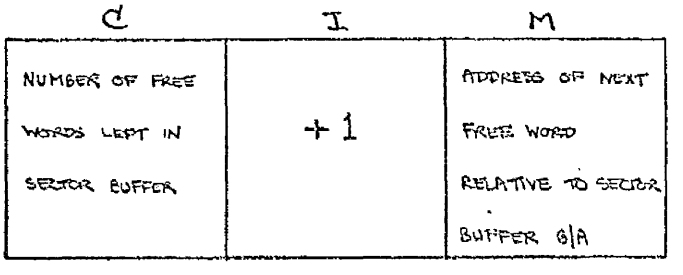


Fig. 8 . Format of sector pack control word.

4.4.3 OUTPUT ORGANIZATION

When a remote job is submitted via CommCon to the GOLD system it is automatically assigned an implicit temporary output file on the CommCon disks. These output files act as an output well under the control of the output control words (OCW1, OCW2) and the interprogram communication block (IPCB). Any terminal directed output (OUT 8, stream -1 or -2) is written to this file sector by sector by Director which also updates DOAW if required by GOAW (see P808 and the IPCB Chapter 4.4.3.1)

When a user submits a job for running he may select immediate output of results to his terminal or else that they be held in the temporary output file system. Since no explicit filing system is provided in the RJE system some method had to be found of identifying his output so that he might inspect it at some later period. The method chosen in this case is to give his output file an identifying code number (which in fact is the disk address of the job heading sector for his job): subsequently, when the user wishes to list the output, he must submit this code to CommCon which checks that it corresponds to an existing file, checks that the job has been run (if not the message "JOB STILL QUEUED" is output on his terminal) and enters the normal output sequence.

GOAW

C	I	M
—	NUMBER OF SECTORS OUTPUT SO FAR IN THIS SEQUENCE	DISK ADDRESS OF OUTPUT FIRST-ON-CHAIN

IF GOAW IS ALL ONES — OUTPUT IS COMPLETE

DOAW

C	I	M
—	—	IF NEGATIVE, FINAL OUTPUT IS AVAILABLE : |M| = TOTAL NUMBER OF SECTORS

IF POSITIVE, NO. OF SECTORS OF OUTPUT AVAILABLE SO FAR

OCW1

C	I	M
NUMBER OF WORDS LEFT IN SECTOR BUFFER FOR OUTPUT	NUMBER OF CHARACTERS LEFT IN CURRENT OUTPUT WORD	ADDRESS OF CURRENT OUTPUT WORD RELATIVE TO SECTOR BUFFER B/A.

OCW2

C	I	M
DISK ADDRESS OF SYSTEM MESSAGE ; IF ZERO, NO MESSAGE	NUMBER OF SECTORS AWAITING OUTPUT ; IF NEGATIVE, NO MORE OUTPUT DUE	DISK ADDRESS OF NEXT SECTOR ON OUTPUT CHAIN ; ZERO, IF NO MORE OUTPUT

FIG. 9. COMM CON OUTPUT CONTROL

4.4.3.1.

There follow short descriptions of a number of routines used in ComsCon output control:

P812: alters the I/O status of a terminal by changing the appropriate bit in V34P899 on or off; in this way V34P899 can act as an output trigger - when it is non-zero there are outstanding output queue demands for terminals corresponding to the raised bits

P808: this is the main output queue control subroutine. Scanning along V34P899 this routine searches for a terminal marked as in a "receive" state. The OCW1 for that terminal is extracted from P809. (See Figure 9). If the counter part is negative then there is no output immediately available and V34 is set to "send" status for that terminal: if it is zero then there is output available but in the next sector on disk which must be read down: fetch OCW2 and check counter part is zero (non-zero means a system message is present); if M-part is zero then all output is finished otherwise check I-part is positive (i.e. one or more sectors of output available in the well) and use OUT#2 to initiate

transfer of the next-sector-on-chain into the correct data sector buffer, suspend output for this terminal (setting DO of DFW1 to indicate transfer called) and leave OOW1 equal to zero and V34P899 set to force inspection of the output requirements of this terminal during the next processing cycle.

Inspection of the coding of the Time Sharing Disk Director KKT80B007UPU revealed that any program initiating a disk transfer was suspended until the transfer was completed: this is because disk transfers are queued within Director and when Director was first written there existed no mechanism for setting lockout on a buffer area due to be involved in a transfer before the transfer started. The PMUQq (set lockout) instruction is available as a field modification and OamCon is written assuming Director has been modified to make use of it. In fact SECOMDIR08UPU, described in Appendix III, does not!

Each subsequent entry to the output routines for this terminal checks lockout on the data buffer and when it clears updates OOW1 and OOW2 from E38 in the buffer (see figs5,9) and recommences output

from the new buffer by removing characters and AND-ing them into the correct fields in the next IOM output buffer set up by P905.

If the I-part of OCW2 is negative, however, then all output is finished, otherwise fetch the GOAW used for this terminal: if it is negative then all output is finished, otherwise fetch DOAW. If DOAW is negative then the absolute value of DOAW gives the total number of sectors available for output from the complete file, and if DOAW is not negative then there are just that many sectors in the file so far and the job is still producing output. If this number ($|DOAW|$) equals the I-part of GOAW then everything available so far has already been output to the terminal and we must wait for the user job to produce more - processing continues for the remaining terminals. If $|DOAW|$ = I-part of GOAW there are more sectors available, GOAW is updated, OCW2 I-part is reset and the cycle recommenced.

This complex process is necessary because Director and CommCon have simultaneous write and read access respectively to an output file and there is no co-ordinating instruction in KDF9 to

read and set a marker in one core cycle (e.g. a destructive load instruction). The GOAW table is CommCon read-write and Director read-only: the DOAW table is Director read-write and GOLD read-only: this means that a program can produce some output and for that output to appear at the users terminal before the program ends, i.e. the basis for user interaction with an active job already exist in the RJE system. The GOAW+DOAW tables are collectively known as the Inter Program Communication Block (IPCB).

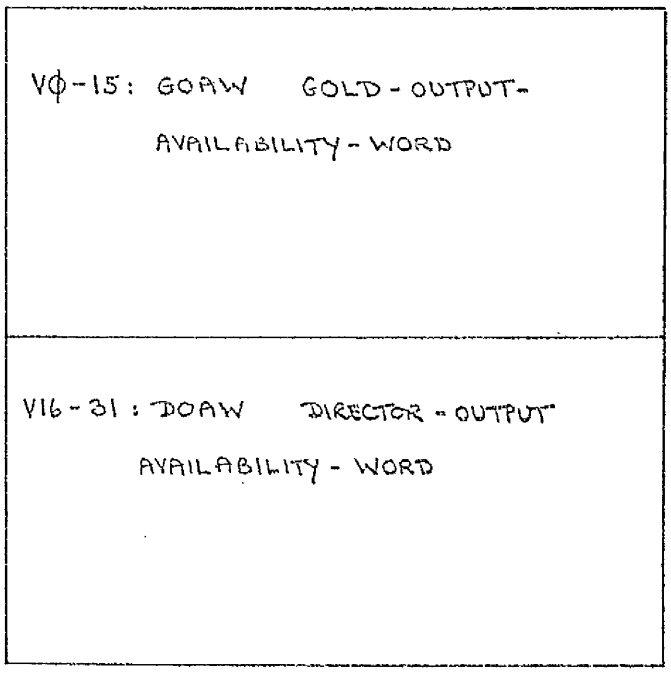


Fig. 10 Internal layout of IPCB

P809: consists solely of a table of 16 V-stores, one pair per terminal, known as the Output Control Words 1 and 2 (OCW1, OCW2) (See Fig 9). Each is in Q-store format and OCW1 contains the information required to access the next string of characters for output from the buffer load in the 40 word data buffer in core, or if this buffer is emptied then OCW2 contains information required to read the next buffer in sequence from disk.

P899: this is the general constants table to which all routines in CommCon refer for details of the configuration of the system (number of terminals, length of buffers, etc.): in this fashion changes can be made very rapidly by altering one common set of constants.

4.4.3.2.

Terminal Directed Output Facilities for User Jobs.

Any remote terminal user wishing small quantities of results returned to his terminal could make use of two extensions of the standard OUT8 facilities.

(i) stream -1 under SECONDIROSUPU allowed upto 16 sectors of online output to be filed in the disk output vell in the implicit output file and output directly or listed later at the users option (See CALL command and prompt "LIST dddddd Y/N:")

(ii) stream -2 must be used following the end of any stream -1 output to close the output file and cause updating of DOAW (see Chapter 4.4.3.1).

A user job must generate stream -1 output in units of one sector or less and for each sector generated DOAW is incremented by +1 starting from zero until stream -2 is detected when terminal directed output must cease and DOAW is negated to indicate that fact to CommCon.

4.4.4. COMMAND STRUCTURE

To exercise and control the facilities of the RJE system a terminal user could issue a number of commands, and I visualized several levels of command. These were

a) special commands - in particular the "½" error correction command which deleted the new FIFO word and typed back the dump and old words.

b) active commands - those which invoke an immediate response from the system e.g. entering a job in the job queue, listing an output file etc. Such commands are always executed directly by CommCon within a

few seconds of being issued.

c) delayed commands - those which are not executed directly by CommCon but by a separate program run under Director by request of CommCon as part of the ordinary job queues. The editor was intended to be such a command program.

d) passive commands - those which cause no directly observable effect as far as the user is concerned but provide optional points of information for the system e.g. a DATA command was proposed which might be used to delimit data strings in a specified fashion. It was not implemented.

Because the system was intended for use by both beginners and experienced users I decided to provide a very elementary form of online documentation via a dual mode of operation at the active command level: the mode was determined by the presence or absence of a fullstop "." character immediately preceding the command terminator "@". If the fullstop was present the command was in imperative mode - the command was executed immediately; if the fullstop was not present the command decoding routines assumed the user was in interrogative mode - he wanted more information on the behaviour of this command: a short descriptive message was then output to his terminal from the disk (see CommCon loader Chapter 4.4.0.1). Commands in interrogative

mode were not executed - the user had to re-issue the command in imperative mode.

There were other types of command available but since they were strictly data for an independant program (e.g. the editor) and not part of CommCon they are not considered here.

Because of very tight limitations on space in CommCon and a rather pessimistic estimate of the capabilities of the disk file a command decoding scheme evolved which ruled out macro-commands in the RDF system. It has to be remembered that CommCon and Director had to perform all the functions of a satellite computer used in other systems in addition to the administration of the round robin, file storage and program execution: real-time demands on the KDF9 were rather heavy and allowing users to create and execute private sequences of active commands seemed too expensive.

In view of experience in 1966 with a paper tape batch processing system using ASR33 Teletypes for tape preparation I decided to make non-printing characters non-significant; this decision and the decision not to do line buffering meant that command delimiters were required ("£" and "@"). Command format was chosen as follows:

⟨command⟩ ::= &⟨name⟩@|&⟨name⟩.@|½

⟨name⟩ ::=⟨letter⟩ | ⟨letter⟩⟨string⟩

⟨string⟩::= ⟨letter⟩|⟨letter⟩⟨string⟩|⟨empty⟩

⟨letter⟩::= A|B|C|..... |Y|Z

Strictly speaking, only the symbols "@" and "½" are commands i.e. symbols invoking special action of CommCon: the character string delimited by & and @ being data to the command decoder.

Certain commands required additional information for their successful execution and issued a sequence of short prompts on the users terminal. In each case the reply was built up in the appropriate FIFO store and extracted upon receipt of a command terminator at the end of the response.

To reduce overheads in Phase One there is a minimal log-in procedure where each sequence of interactions with the system begins with a &NEW.@ command and ends with a job being placed in the job queue.

Note that there is no "command state" in the RJE system - a command may be issued at any time under any circumstances.

4.4.5 COMMAND HANDLING

Upon completion of processing of the KCM input buffer V2P899 is OR-ed with V3P899 in N1: a non-zero result indicates that a command terminator was detected in the last input buffer (V2P899) or that a suspended command awaits completion (V3P899) and P821 is entered. P821 searches the DFW Table for bit D2 in DFW1 of any terminal: P821 is thus only entered when required. Upon finding D2 set, the corresponding terminal bit in V3P899 is tested and if set the suspended command numbered in D5-7 of DFW1 is entered directly - if D3 is set it indicates that the FIFO of the corresponding terminal contains the users reply to a prompt; if D3 is not set the FIFO holds a command name.

P822 extracts the command name by searching the FIFO from the dump word through the old word to the new word for the initial & symbol: after finding this all characters up to the @ are taken as a command name and saved in P899 temporarily. All characters preceeding and following the name string in the FIFO are copied into the sector buffer and the FIFO is reset. Failure to find the & or @ symbols causes an error exit to P8h1 which warns the user.

If P822 ends successfully P823 is entered to

check the name: firstly the last character is checked to be a fullstop (imperative mode) and the least significant character proceeding this is used in a form of table look-at to find and enter the command coding. If the fullstop is absent (interrogative mode) then the descriptive message about this command is loaded from disk and output to the users terminal.

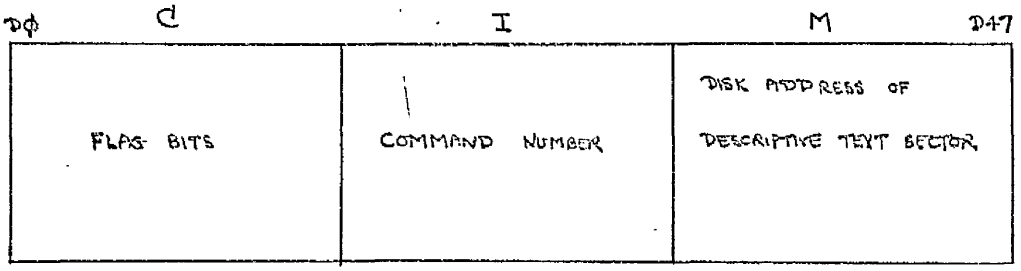
4.4.5.1.

There follow short descriptions of a number of routines used in CommCon command handling:

P822: this subroutine scans the FIFO for a given terminal and extracts a command name delimited by £ and @. (See above)

P823: uses the command name to search a command dictionary in P897 after checking for special terminators ! " $ % & ' (unused in RJE system) and interrogative mode. The least significant character of the name is used to reference the base dictionary e.g. all commands ending in "D" fetch the fourth entry, which is a pointer to the head of a linked list of command names. This list is searched using the rest of the command name as a mask; an unsuccessful match fetches a link to the next list entry and so on until a zero link is

found indicating the list is exhausted and the command is invalid: successful matching is followed by the extraction from the dictionary of a command parameter word of the format shown in Fig 11.



NB. THE COMMAND NUMBER IN I-PART (USED IN P824) DIFFERS FROM THE INTERACTIVE COMMAND NUMBER USED IN P821.

Fig. 11 Layout of Command parameter word in command dictionary.

D1 of the flag bits indicates that the preceeding data file on disk is required during execution of this command, and ensures that the file is correctly chained and terminated.

P824: this routine acts as a switch using the command number from the I-part of the command parameter word to enter the command coding

P828: inspects D1 bit of the command parameter word and if set saves the current contents of the 40 word data buffer for this terminal to disk. It then reads the command descriptive message into the buffer by placing the message address in the C-part of 0042 for use by the output routines (See P806).

If D3 is set when the "@" character is detected then the FIFO contains the reply to a prompt: D5 to D7 of DFW1 hold the interactive command number which acts as a switch to enter the appropriate command at a side entry (the "running" entry) to deal with a response to a prompt.

4.4.6 COMMAND CODING

All commands are written as serially re-entrant pure procedures as part of the general segmentation plan. Where necessary, all commands may suspend execution on behalf of one terminal and start on behalf of another. There are two reasons for suspension:

a) the command routines are waiting for further data from the terminal (e.g. reply to a prompt)

or b) access to a buffer area is prevented by lockouts.

The former uses a suspension marker and prompt counter in B11 of the data buffer (see Fig. 12.) whose value is used as a switch to enter the command coding at the end of each user response to the preceeding prompt (see §4.4.5.; D3 of DFW1 is set one when a prompt is issued.) The latter uses two markers: a bit in V3P899 corresponding to the terminal address (which when OR-ed with V2P899 forces re-entry to P821), and D0 and D2 of DFW1 which are set one when a transfer has been started and a command is outstanding respectively; e.g. if D_n of V3P899 set one, D0 of DFW1 setzeno for terminal n and D2 of DFW1 set one for the same terminal, then the command whose number is in D5-D7 of $DFW1_n$ is suspended due to lockouts and a transfer it wishes to initiate on its own behalf is unstarted. The outstanding bits set in V3P899 force re-entry to P821 after each IOM

transfer cycle and P821 includes a test of V3P899 for each terminal: if a bit is raised in V3P899 corresponding to a particular terminal then a command routine working for that terminal has suspended itself due to lockouts on the terminal data sector buffer: the interactive command number is found in DFW1 D5-7 of this terminal and is used to re-enter the command coding directly. Each command is responsible for determining whether or not it can continue processing due to lockouts - if not it simply exits again leaving all markers set to force re-entry once more following the next IOM transfer.

All commands may use the 40-word data buffer for each terminal as workspace on behalf of that terminal - any user data there is saved in the implicit input file if D1 of the command parameter word is set 1.

Note that the interactive command number in D5-7 of DFW1 acts in a switch to re-enter the command after suspension and differs from the command number in the command parameter word which acts in a switch to enter the command initially.

4.4.6.1.

There follows brief descriptions of the coding of each of the commands implemented in the RJE system.

P840: Command 0 - NEW

This command initiallizes CommCon for the next job submission from the users terminal: it allocates a new implicit input file, puts the terminal in receive status in V34P899, resets the FIFO store, clears DFW1 and enters the address of the first-sector-on-chain of the input file in the I-part, the chaining address sequence into DFW2, resets the SPCW, clears GOAW and marks the terminal as "active" in V49P899. A message indicating the time, date and system edition is printed on the users terminal and the system is ready to accept the next job. This command always runs to completion, issues no prompts and has no reason to be suspended.

P841: Command 1 - CALL

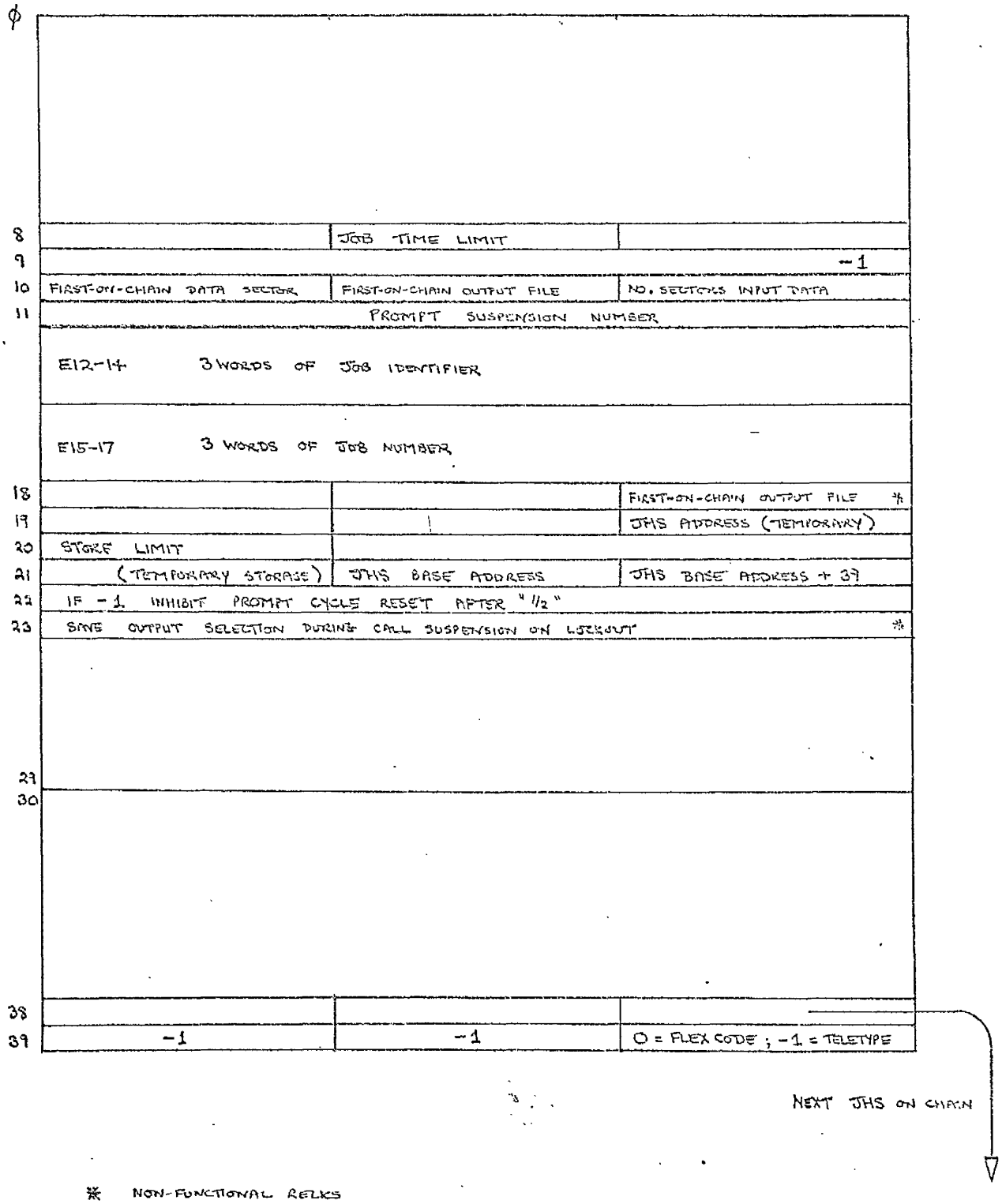
This command is used to call for the running of a named binary program: the command issues a series of prompts on the terminal to which the user replies with the required information e.g. job identifier, job number, time limit, store limit and output option.

The main entry to P841 sets the interactive command number 0 in D5-D7 of DFW1 and sets D3 one (i.e. reply to prompt expected). The prompt suspension marker (see 4.4.6.) in E11 of the

data buffer and the prompt reset inhibit marker in E22 are both cleared. Control then passes to the main side-entry, where the prompts are dealt with, which updates the value of E11 and uses the original value to enter the coding to deal with the reply to the previous prompt, repeat the prompt if the reply is invalid or issue the next one.

Information from the user is combined with information from CommCon to create a 40 word Job Reading Sector to be entered into the job queues. (See Fig.12.).

Each JRS carries pointers to its chained temporary input and output files. The final entry to P841 causes the prompt "PRINT dddddd Y/N;" to appear on the terminal: the six digit code number is a disk address informing the user where the JRS is to be queued on disk. If the user replies "Y", GOAN is set up to allow terminal directed output to appear immediately the job starts to run; if he replies "N", it is assumed that he wishes the output from the job held in the output file only. To retrieve this output the user must use the LIST command, quoting the above six digit code (See LIST command).



* NON-FUNCTIONAL RELICS

E15 is used by OLOFYDE1 to store PROTAPE identifier for a PCL job

FIG. 12. STRUCTURE OF JOB HEADING SECTOR

FOR USE BY JOBORG --- UPU.

On completion of assembly the JHS is written to disk in the next queue position and an OUT-4 used to initiate or update the Director job queue pointer for the CommOn program stack; finally GOAW is set up as requested above.

Because of the way the prompts appear in the CALL command, typing "½" in response to the prompt "PRINT dddddd Y/N;" does not reset the prompt counter to zero and cause all the prompts to be re-issued. This is because a JHS address cannot be assigned to a user job until the last possible moment (to avoid increasing the load on JOBORG): in fact the address is assigned only during assembly of the "PRINT dddddd Y/N;" prompt where it appears as the six digit code - if the prompt cycle was restarted from the beginning at this point yet another JHS address would be assigned. Such a possibility is avoided by the use of a prompt reset inhibit marker set in E22 of the data buffer where the JHS is being assembled - if E22 is zero then typing "½" restarts all the prompts of the command afresh: if non-zero then only the last prompt is repeated. E22 is set non-zero immediately prior to the prompt being issued.

P842: Command 2: PROMPT

This is really a side entry to command 1, CALL, where the job identifier, number, time and store limit are known: it results in the placing of a call for PROMPT in the job queue to perform some file maintainance operations. The parameter flag D3 of DFW1 for the terminal is set one, D5-7 of DFW1 is also set to 2 (interactive command number 2 - PROMPT) and the prompt counter in W1 of the data buffer is set to 5 (so that the first prompt issued in response to the command is "PRINT dddddd Y/N;" Control then passes to P841 (CALL) at the main side-entry.

P843: Command 3: LIST

Sets interactive command number 1 in D5-D7 of DFW1 and the parameter marker D3 for this terminal and issues the prompt "LIST FILE NUMBER;" to which the user should respond with the same six digit code number issued during the CALL or PROMPT command interaction. The code number (disk address of the JMS) is checked for range in P814 and correspondence to an existing file in P843 and used to read down the JMS from disk. P843 may suspend itself here until lockouts are cleared, by use of V3P899 and D0 of DFW1 for this terminal

(sec 4.4.6.). On resumption E9 of the loaded JHS is inspected: if the C-part is zero then the job has been completed and output is available - GOAW is set correspondingly and the output-first-on-chain used to get the first sector of output data for listing. If the C-part is non-zero then the job is still queued and a message to this effect is output to the user.

P897: Command dictionary

V0 to V6 form the base dictionary: each 12 bit field corresponding to a given letter is the pointer to the head of a linked list of commands whose names end with that letter. This list is held in the main dictionary.

V7 to V46 is the main dictionary with space for ten command names; each entry consists of four words as shown in Fig. 18.

The first two words comprise the right justified command name less its final character as used in the matching process. The third word contains a link address to the next list entry corresponding to that letter, and the fourth word contains the command parameters as shown.

Considering the number of commands likely to be available and the frequency of their use in a

RJE system such a crude table look-at is unnecessarily slow and wasteful of space: it was implemented purely out of interest.

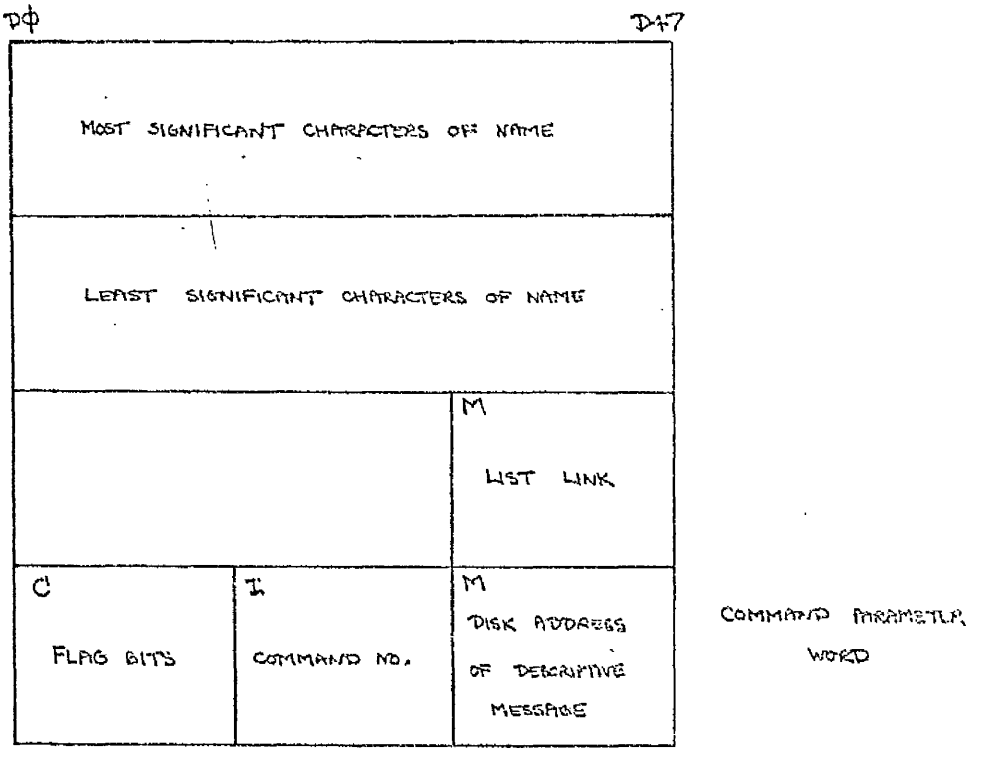


Fig. 13. Structure of command dictionary list entry.

CHAPTER 5. Teletype PROMPT, GA2OOOO--US4.

PROMPT, KAZOOA7---US4, is a standard KDF9 segmented system for the establishment, maintenance and translation of users programs using the disk file as a random access storage device by means of a set of privileged OUTs. PROMPT's role as a program testing scheme is of great advantage in a RJS system and we made full use of this in Phase One, both as a development tool and as a major subsystem when modified.

Like the other components of the GOLD system, PROMPT runs in program mode but can be considered in many ways as an extension of Director, being able to indirectly modify Director's ASL (disk storage map) and access reserved areas of the disk file. PROMPT consists of three basic sections.

a) resident program with phase control (overall control) and file control (I/O and Director communication)

b) segment area (for Usercode compiler, amender, WALGOL compiler, disk updater etc)

c) liason area (intersegment communication and constants storage)

Between them, PROMPT and Director also provide a basic file manipulation and storage system for program texts

and compiled code.

To provide these facilities for remote terminal users a modified version of PROMPT was produced; running as a remote job, data input was by trapped reads from the implicit data file associated with its JRS: all output was converted to teletype character code by my version of Mr. Bowie's code conversion package in file control. This package was modified to recognise the directive END PROMPT (which shuts down the system) and to output the final buffer via OUT8, stream-2 in order to close the output file.

PROMPT's five hole I/O conversion tables were removed, the five hole tape I/O option invalidated and the space saved used for the new OUT8 buffers and code conversion tables.

File control's handling of all I/O on behalf of each segment made modifications to the output routines much simpler, and phase control's method of segment identifier construction allowed the running of all standard segments in Teletype PROMPT simply by changing the first character of their individual identifiers from K to G, and recompiling to a slightly increased base address to allow for the code conversion package in the resident section.

The structure of KAZ81A7--U34, the amender segment, was analysed with a view to insertion of character manipulation facilities but these were not implemented.

CHAPTER 6 INPUT WELL MAGNETIC TAPE SYSTEM

The scheme was designed to accumulate the input character strings from each terminal in a series of buffers on a common magnetic tape. When sufficient data had accumulated to make a disk access "worth-while" or when a terminal user issued a command which required the preceeding character data string then the tape was rewound and the contents of the tape blocks sorted to disk. The basic structure of communications controller for the Input Well Magnetic Tape (IWMT) system was similar to the all-disk version described above but for the use of the magnetic tape.

To initialise the IWMT system a B-type loader program claimed two disk platters from Director, claimed the ICM channels and cleared the ICM, claimed a type 60 magnetic tape for use as the input well, rewound and relabelled the tape <ZERO> and positioned it after the label block. The loader also loaded a set of command descriptions (for interrogative mode use of active commands) to disk, generated the current date in teletype characters, cleared core and loaded CommCon by an OUT+1 with the date in N1,2; the type 60 tape device number in N3; ICM output device number in N4; and ICM input device number in N5.

The IWMT version of CommCon contained space in core for a single 128 word magnetic tape buffer; this

buffer was divided into two fields (See Fig. 14).

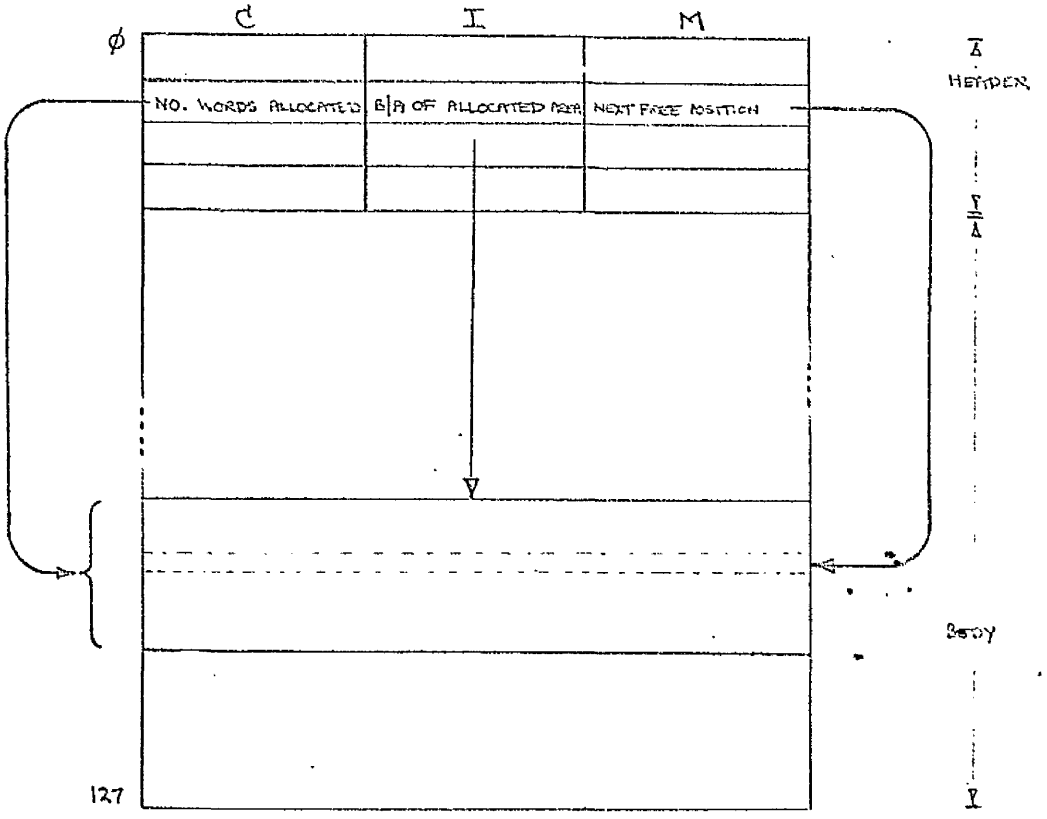


Fig. 14 Divisions of IWMT buffer for four terminals.

Each header word had the format shown and was used by P802 to store the FIFO dump word into the buffer body then suitably updated: when the counter became zero P803 was entered to repack the buffer contents. This was done by finding the terminal header word showing the most unused space (highest count), reducing its allocation to some minimum amount and moving the freed space onto the end of the exhausted allocation by shifting all the other relevant core areas up or down within the buffer, updating the base address and next address values in affected header words as necessary. The allocation of the exhausted terminal header was increased by the amount freed and processing continued.

This procedure ensured that terminals transmitting at a higher speed were assigned more storage in the buffer than those transmitting at a lower speed and also packed the buffer tightly.

When the amount of space to be gained by this method fell below a minimum specified in P899 the buffer was written to the IWMT preceeded by a one word marker containing a block count. Rather than reset the headers to some standard value they were reset to reflect the relative core allocations that prevailed within the buffer just prior to that moment, coping with changes in terminal input rates in a dynamic fashion.

If a terminal user issued a command that required the character data string preceeding, the final core block was written to the IWMT and the tape sorted to disk. The sort read back the tape data blocks one by one into the same core buffer as for output, and extracted the data for a terminal, packing it into a disk buffer and chaining to disk. A separate pass along the tape was made for each terminal, backwards for terminal 0, forwards for terminal 1, etc., until all data had been removed from the tape and copied to disk in chained files. The theory behind the separate pass for each terminal's data was that it meant only one disk seek per terminal and not one per disk transfer.

Output under this system was again via a disk output

well, and, of course, used one 40 word disk sector buffer per terminal. The facts that this space was permanently assigned to its terminal, that it was only used for a fraction of the time it took a user to type his input and that the IWMT proved slow, clumsy and inefficient were obvious early in 1967 and the scheme was abandoned in September of that year.

Our initial enthusiasm for the idea was based on the following calculations: assume user types in a 3 page program at 50 lines per page and 30 characters per line, i.e., 4,500 characters; a 128 word block contains about 1000 characters - assume user has typed 200; therefore a program needs 20 blocks on tape at 400 chars./inch occupying about 8 feet - with only four people active at terminals the average length would be about 2 feet: this could be sorted in about 1 second while an IOM transfer was running.

Of course, it is now obvious that users just do not behave in this fashion and prefer to submit bulk input as tape or cards used to create a named file, and then edit and amend this file in an online session involving further small quantities of data and frequent commands: experience also showed that Director's disk access mechanism was inadequate (see page 37). Eyeball timing of the sort indicated it took over two seconds for only 4 terminals with just a few blocks on tape - and the more terminals that

are active the more frequently will the sort be required: it does not really save anything.

CHAPTER 7 MIDAS

It had been the intention of the Department to demonstrate the initial capabilities of the RJE phase of the GOLD project in the week of June 5th, 1967. Unfortunately the reliability of the KDF9 disk file had become so low by the last week in May that we had to discontinue using it. In order to avoid cancellation of the demonstration we rewrote the IWMT version of Phase One to handle a single terminal using one magnetic tape for temporary input file storage and a second tape for temporary output file storage. MIDAS was restricted to the Whetstone Algol Interpretive compiler and the two commands RUN and NEW and was designed, written and largely debugged in ten days. Mr. Bowle was responsible for the modifications to WALGOL.

MIDAS operated as follows: following a NEW command, input character strings comprising a WALGOL program were converted from teletype code to flexowriter code and accumulated on a type 60 magnetic tape. When a RUN command was issued the type 60 was deallocated, a privileged OUT was used to load the modified WALGOL compiler to a free level and after a long "noise" message MIDAS suspended trying to claim a type 61 magnetic tape. When the modified WALGOL compiler entered core it claimed the type 60 tape as the input device, a type 61 as its output device and read

down the program typed in from the terminal, translated and ran it and output all diagnostics and results to the type 61 magnetic tape which was then deallocated and immediately claimed by the suspended MIDAS communications controller. The WALGOL compiler then deallocated the type 60 tape and ended. CommCon read down the output, converted it from flexowriter code to teletype code and output the results to the terminal, deallocated the type 61 tape, claimed the type 60 and prepared to accept the next job.

To the terminal user MIDAS looked like a restricted version of Phase One but for the long "noise" message designed to cover up the dead time while WALGOL entered core and claimed the type 60 tape. Severe electronic noise problems on the ICM reduced MIDAS' effectiveness even further.

CHAPTER 8 CONCLUSIONS

When the project was abandoned in November 1967 we had four terminals fully operational at which it was possible to establish, amend, compile and run a job, either getting the results back directly at the terminal or by listing the implicit output file; we had an efficient, modular operating system 50% of which ran in program mode and was fairly easily tested and debugged: this operating system supported full multiprogramming in a fashion which promised efficient use of the KDF9; and finally we had a much clearer concept of what was needed in the next phase of development. Measured ratio of elapsed time to run time for CommCon supporting four terminals was approximately 20:1.

From a personnel point of view, the almost immediate loss of 50% of the team when we started imposed a heavy strain on our limited resources - it left two people to design, investigate, code and debug all software, design and develop all special hardware and, in the case of Mr. Matkovits, also be responsible for day-to-day engineering support for the KDF9: this, and experience early in 1967 of persistent hardware problems with the newly installed disk file, probably did more than

anything else to force cancellation but for a comparatively small cost (less than $1000 for hardware) the Department gained experience not to be had in any other way.

I think our views of the capabilities of the disk file were probably unduly pessimistic and had a bad influence on the way that my section of the project, in particular, developed.

Reviewing my own parts of the project three years after we started work it is clear that there are several major areas of weakness in the design of CommCon and in its relationship to the rest of the system

(i) segmentation of CommCon should have been implemented right from the start, not kept in mind as a later modification

(ii) the ad hoc filing system that developed in lieu of Mr. Bowie's part of the project was totally inadequate, lacking long term storage of named data files and even any form of space return mechanism although the latter could have been trivially added

(iii) the choice of 40 word disk sectors as the standard buffer size and the effect of this on CommCon due to the 32 word lockout control was not properly taken into consideration when work began

(iv) subroutine modularity has been allowed to breakdown in the command coding (but could be corrected very easily)

(v) the overall appearance to the user is awkward and unsuitable for general use - in particular the NEW command is unnecessary and could be omitted with suitable modifications to P821

(vi) reference to P900 to allow software interrupts should probably occur more frequently, in particular within P821 and P808

(vii) efforts to keep CommCon small and fast were overdone - the result is not as flexible as it should have been

The utility of Phase One as an implementation tool for Phase Two (the interactive PDP-8 based multiaccess system) is, of course, unproven as is its utility as a general user system but I feel that it could well have sufficed for the former. However, it is my personal opinion that the interfacing of a simple real-time device like the ICM directly to the KDF9 was not the best way to provide a RJE system - too much time consuming overhead results. This overhead is largely of a character handling nature and not the kind of task best suited to the capabilities of the KDF9. The flexibility and low cost of

many small computers available on the market today provide a superior solution to the RJE problem: the ICM on KDF9 could well have provided the basis for a "multiplexor channel" in the IBM sense for slow devices and it is in this role that development might continue. The successful development of two contrasting systems based on a PDP-8/KDF9 configuration (the TSDD-based MUDON system of University of Leeds, and the EGDON-based COTAN system of UKAEA Culham) suggests that but for its unfortunate early history the GOLD System might have gone on into Phase Two as a successful system.

APPENDIX I.

CommCon occupies 992 words of compiled code and V-stores (including 89 V-stores for the dummy L1 routine) and 960 words of X-stores for buffers etc.

INDEX TO FLOW CHARTS

Communications Controller Loader

CommCon

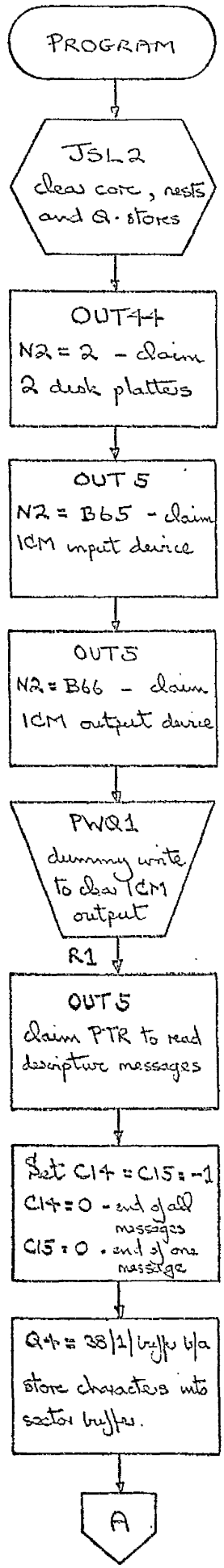
Coding and Flow charts of Communications Controller

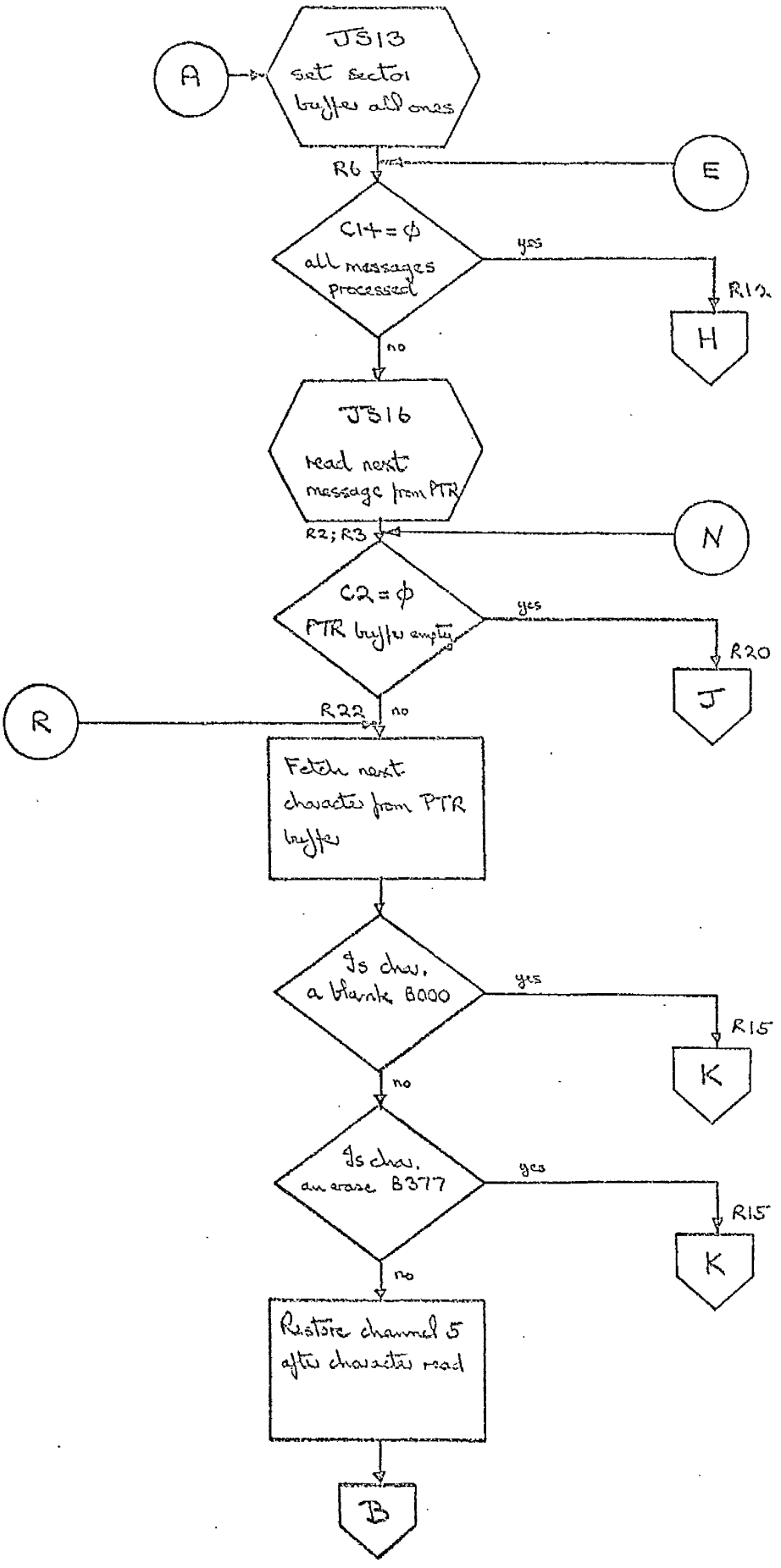
Loader

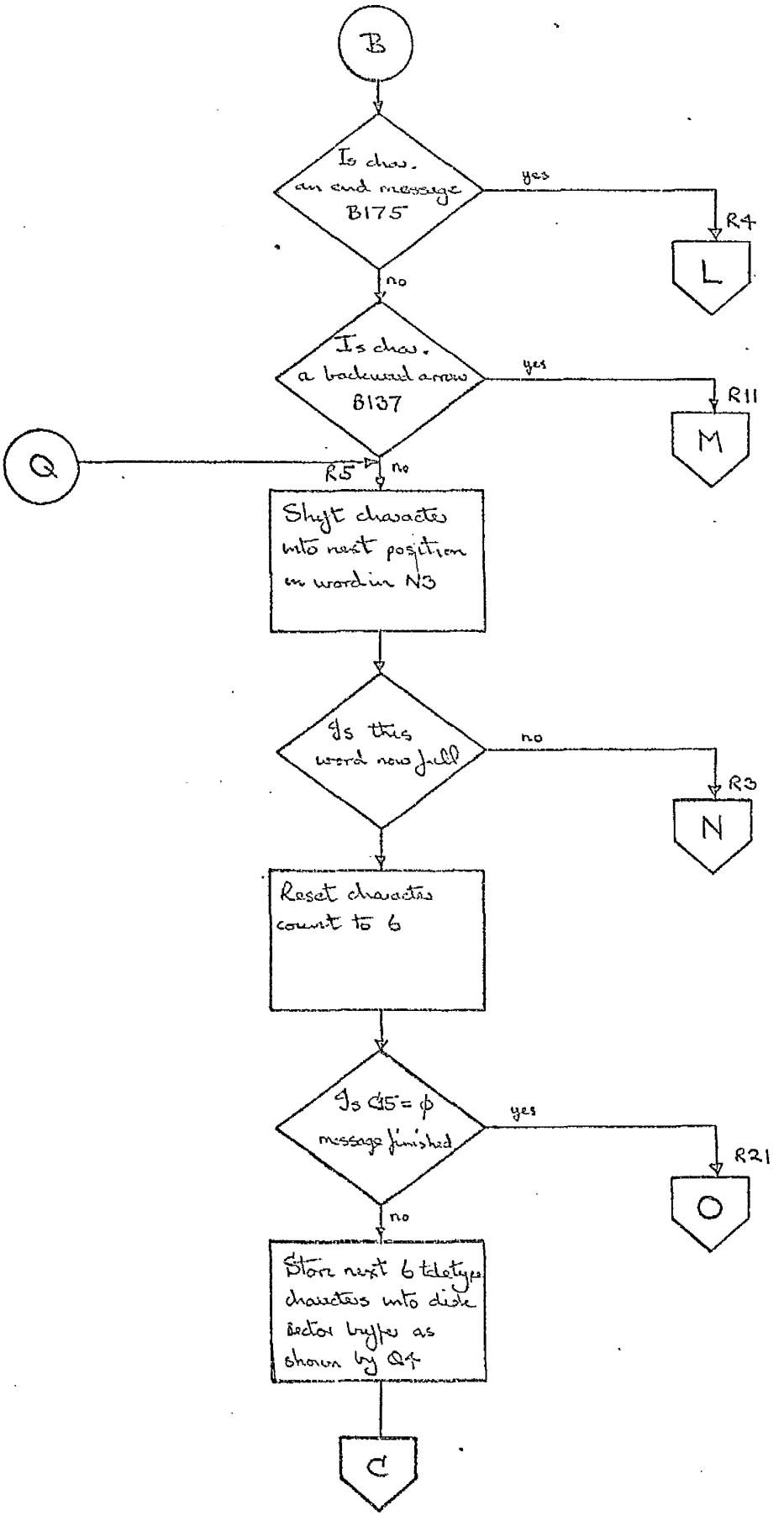
```
ST 2000;
TL 7200;
V23;
PROGRAM;
        V1=Q 0/AY0/AY227; (CHAR READ BUFFER);
        V2=Q 0/AY256/AY295; (SECTOR BUFFER);
        V3=B 1/15; (FIRST FREE SECTOR FOR WRITE);
        V4/5=P*9NC*UD*GC*UCOMMCON--UP1;
        V6=B 240 5012 0240; (4 TELETYPE SPACE CHARS);
        V7=Q 40/1/AY256;
        V13=Q 0/AV14/AV17;
        V14/17=P*Q8S*U£DISC£WRITE£PARITY£FAIL;
        V18=Q 0/AV19/AV22;
        V19/22=P*Q8S*U£MESSAGE£PARITY£ON£READ;
        V23=Q 38/1/AY256; (DISK SECTOR BUFFER);
        JSL2;
        SET 2; SET 44; OUT;
        SET B65; SET5; OUT; (INPUT);
        SHL+32;
        SET B66; SET5; OUT; (OUTPUT);
        DUP; =C1;
        SET AY0; DUP; =I1; =M1;
        PWQ1; (SET UP DUMMY READ TO CLEAR CARD PUNCH BUFFER);
        SHL+32;
1;        (LOAD MESSAGE SECTOR);
        SET2; SET5; OUT; (CLAIM READER);
        SHL+32; V1; OR; =Q1;
        ZERO; NOT; DUP; =C14; =C15; (SET MARKERS);
        V23; =Q4;
        JS13; (SET UP DISK BUFFER);
6;        J12C14Z; (ALL MESSAGES PROCESSED);
        JS16;
        ZERO; NOT;
2;        ZERO;
3;        J20C2Z;
22;         MOM2Q; DUP; J15=Z; (J BLANK);
        SET B377; J15=; (J ERASE);
        SHLD=4; DC3;
        SHC=2; SHLD=1;
        SHC+3; SHLD=2;
        SHL=1; SHLD+7; (CHAR INM1/0/PACKED WORD);
        SET B175; J4=; (END OF BLOCK);
        SET B137; J11=; (END OF ALL MESSAGES);
 5;       CAB; SHL+8; OR; (PACK/CHAR/0);
        REV; J3C3NZ; (0/PACK);
        SET 6; =RC3; REV;
        J21C15Z;
        =MOM4Q; (STORE 6 PACKED TELETYPE CHARS);
        NOT;
        J2C4NZ;
21;       ZERO; NOT; REV;
```

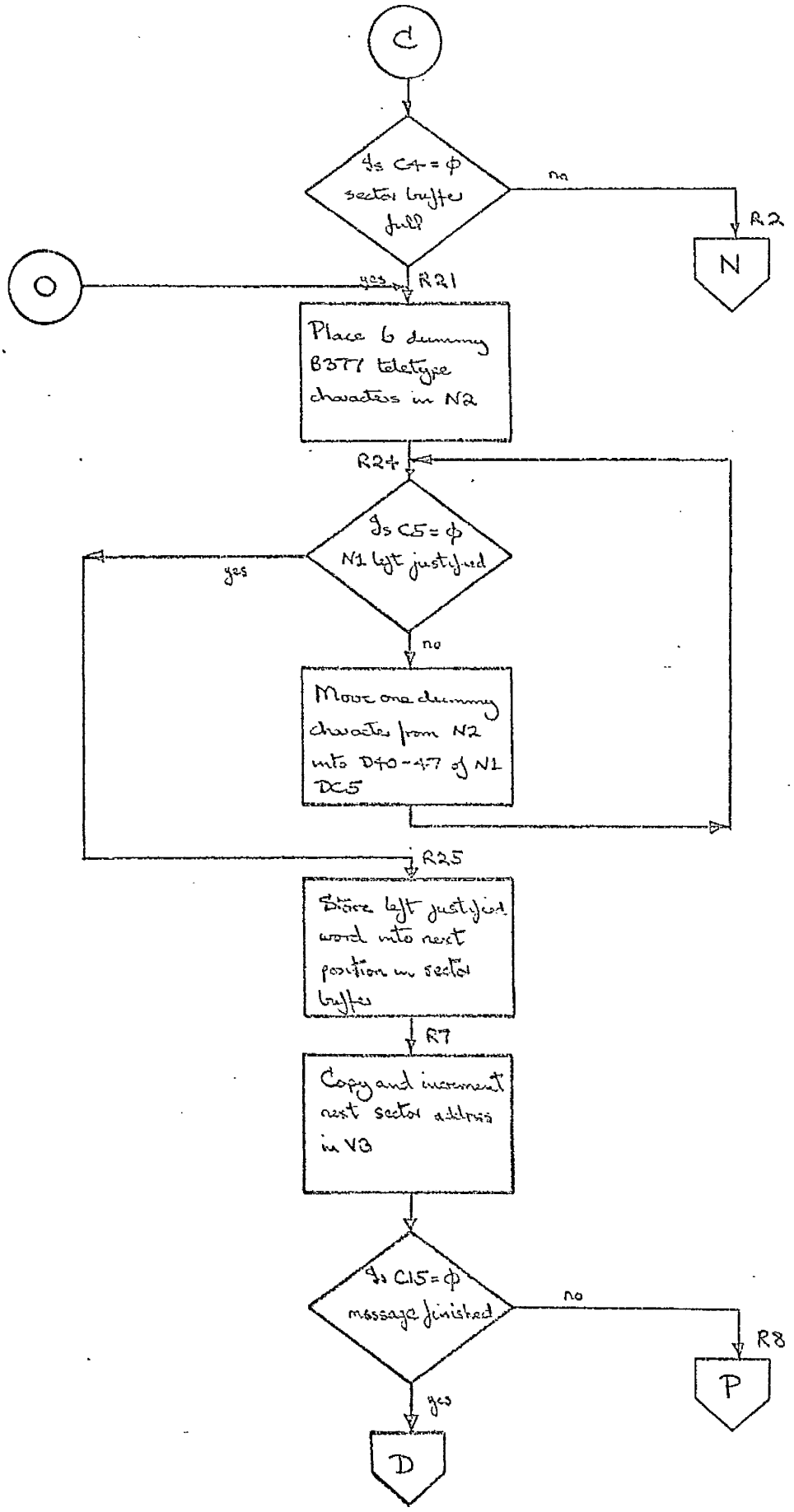
```
25;       =MOM4Q; ERASE; ERASE;
 7;       V3; DUP;
         SHL-32; NOT; NEG; SHL*32;
         =V3; (INCREMENT TO NEXT SECTOR);
         J8C15NZ; ZERO;
 9;        SET 38; C4; =;
         SHL+16; OR;
         =Y294; (Y294=0/NO.ACTIVE WORDS/NEXT SECTOR);
10;       V2; OR;
         SET 41; OUT; (WRITE UP SECTOR);
         V23; =Q4;
         SET 47; OUT;
         J17TR;
         JS13; (CLEAR DISC BUFFER);
         ZERO; NOT; =C15;
         (PREPARE TO RECEIVE NEXT MESSAGE);
         J6;
16;       PIDQ1;
         PARQ1; J18TR;
         I1; =RM2;
         SET 228; =C2;
         SET 6; =RC3;
         EXIT 1;
11;       ZERO; =C14;
 4;        C3TOQ5; (SAVE FOR PACKING);
         C0T0Q3; C0T0Q2;
         ZERO; =C15; (END OF BLOCK);
         ERASE; SET B377;
         J5;
 8;       V3; SHL-32;
         J9;
15;       ERASE; J22C2NZ;
20;       JS16; J22; (RELOAD BUFFER);
12;       C1; SET 6; OUT; JSP1;
         JS3L2;
         V5; V4;
         SET 1; OUT; (ENTER COMMCON WITH N1,N2 = DATE IN TELETYPE
         CHARACTERS; N3 = ICM OUTPUT DEVICE NUMBER; N4 = ICM
         INPUT DEVICE NUMBER);
19;       SET 8; OUT;
         ZERO; OUT;
17;       V13; J19;
18;       V18; J19;
13;       V7; =Q7;
£23;       ZERO; NOT; £=MOM7Q; J23C7NZS;
         EXIT 1;
L2;       (CLEAR WORKING STORES)
 1;        J3 EN
 2;        ERASE; J2 NEN
 3;        LINK; J5 EJ
 4;        LINK; ERASE; J4 NEJ
```

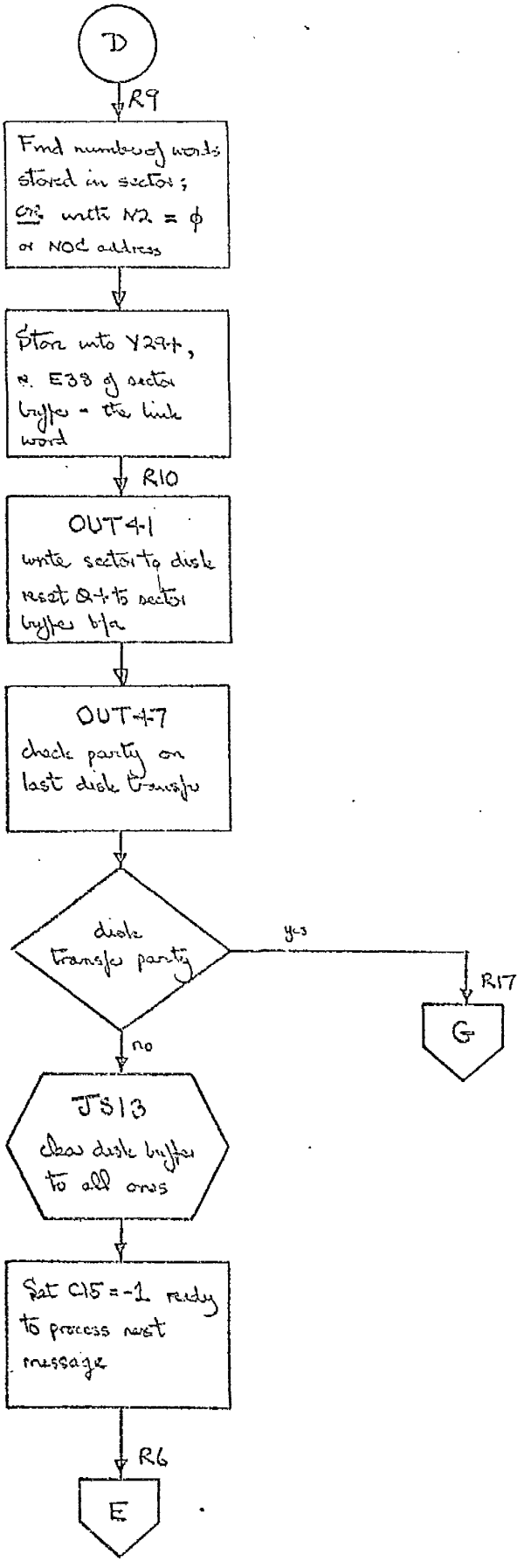
```
        SETAVOPO; SET8; DUP; =RH1; =; =C1; J7
£6;       ZERO; =MOM1Q
£7;       J6C1NZS
        MOMON; MOMOHN; REV; SETAWO; DUP; =RH1; =; =C1; J9
£8;       ZERO; =MOM1Q
£9;       J8C1NZS
        QOTOQ1; QOTOQ2; QOTOQ3; QOTOQ4; QOTOQ5;
        QOTOQ6; QOTOQ7; QOTOQ8; QOTOQ9; QOTOQ10;
        QOTOQ11; QOTOQ12; QOTOQ13; QOTOQ14; QOTOQ15;
        ZERO; =TR; VR; EXIT1
        (END OF L2);
P1; (GET DATE FROM E7);
        SET 8; =RC6; (COUNT FLEX CHARS);
        SET 6; =RC7;
        ZERO;
        E7; (DATE IN FLEX CHARS);
2;        J5C6Z; ZERO; SHLD+6; SET B40; OR;
        SET B57; J1=; (TEST FOR / );
        DUP; BITS; SHC-1;
        J1*LTZ;
4;        DC6; CAB; SHL+8; OR; DC7; REV; J2C7NZ;
        SET 6; =RC7;
        ZERO; REV; (E7/NEW PACK WORD/OLD PACK WORD(FIRST 6 CHARS)
        J2;
5;        (E7 EMPTIED);
        ERASE; SHL+32; (MOVE UP LAST TWO CHARS);
        V6P0; OR; REV;
        EXIT 1;
1;        SET B200; OR; J4; (PARITY);
FINISH;
```

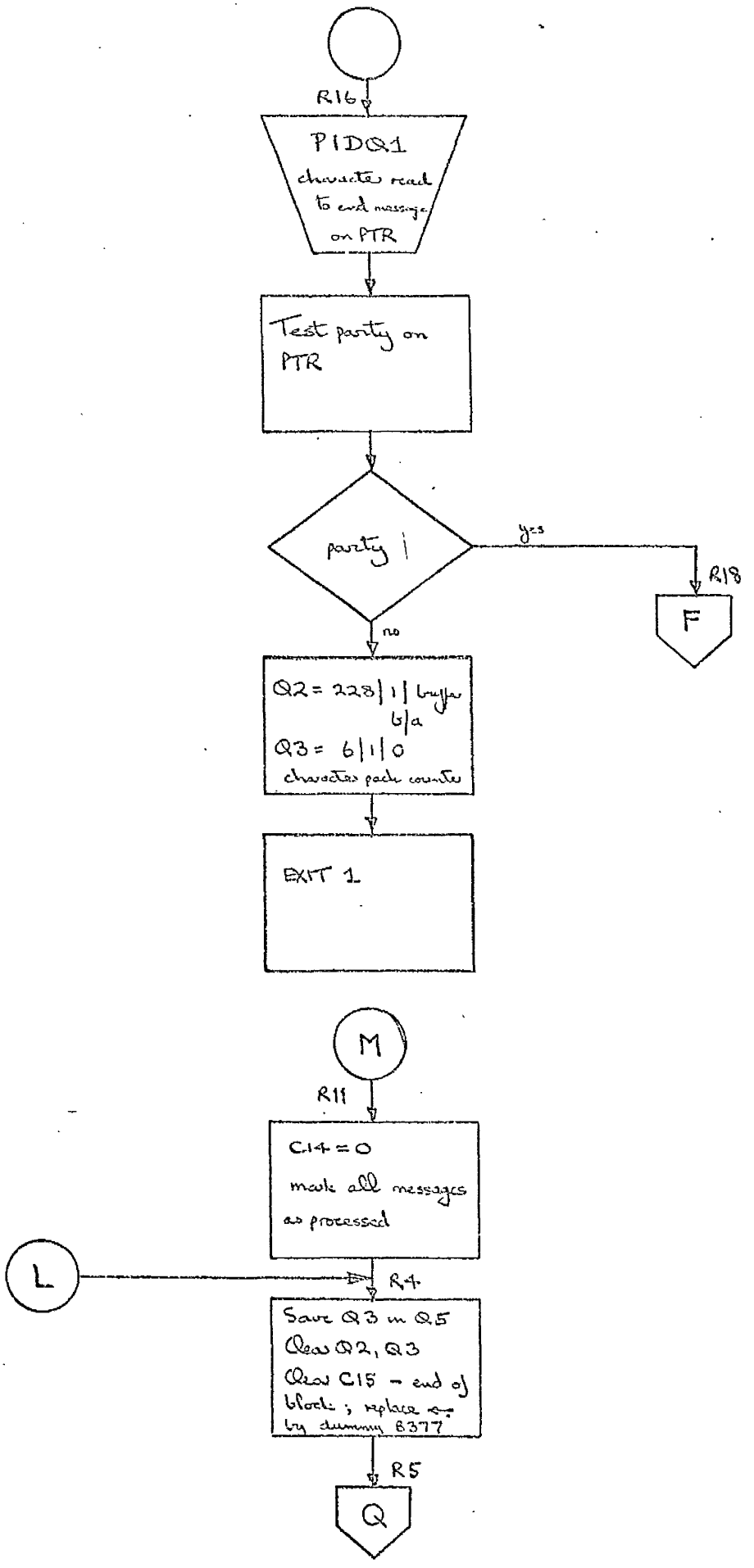

PROGRAM
JSL 2
clear core, nests and Q. stores
OUT 4
N2 = 2 - claim 2 disk platters
OUT 5
N2 = B65 - claim ICM input device
OUT 5
N2 = B66 - claim ICM output device
PWQ1
dummy write to clear ICM output
R1
OUT 5
claim PTR to read descriptor messages
Set C14 = C15 = -1
C14 = 0 - end of all messages
C15 = 0 - end of one message
Q4 = 38/1/ buffer b/a
store characters into sector buffer.
A

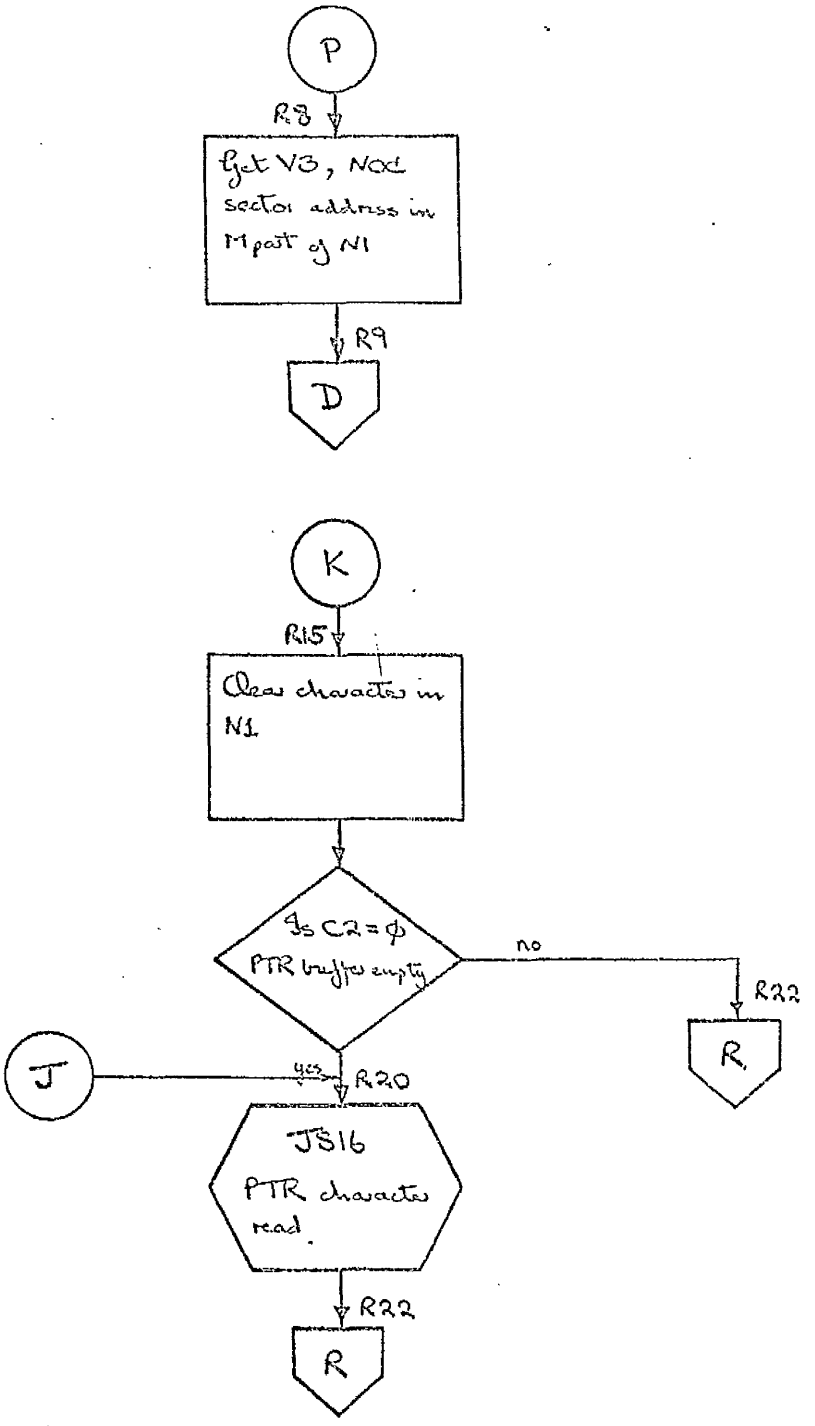

A
JS13
set sector buffer all ones
R6
E
C14 = ϕ
all messages processed
yes
R12
H
no
JS16
read next message from PTR
R2; R3
N
C2 = ϕ
PTR buffer empty
yes
R20
J
R
R22
no
Fetch next character from PTR buffer
Is char. a blank B000
yes
R15
K
no
Is char. an erase B377
yes
R15
K
no
Restore channel 5 after character read
B

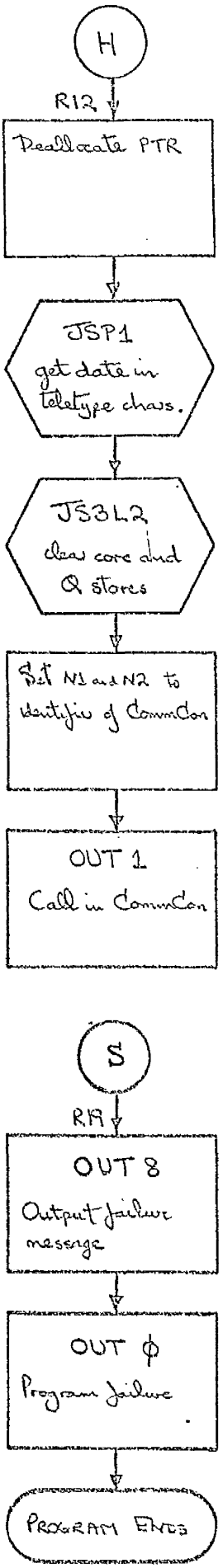

B
Is char. an end message B175
yes
R4
L
no
Is char. a backward arrow B137
yes
R11
M
Q
R5
no
Shift character into next position in word in N3
Is this word now full
no
R3
N
Reset character count to 6
Is Q5 = ϕ message finished
yes
R21
O
no
Store next 6 teletype characters into disk sector bytes as shown by Q4
C

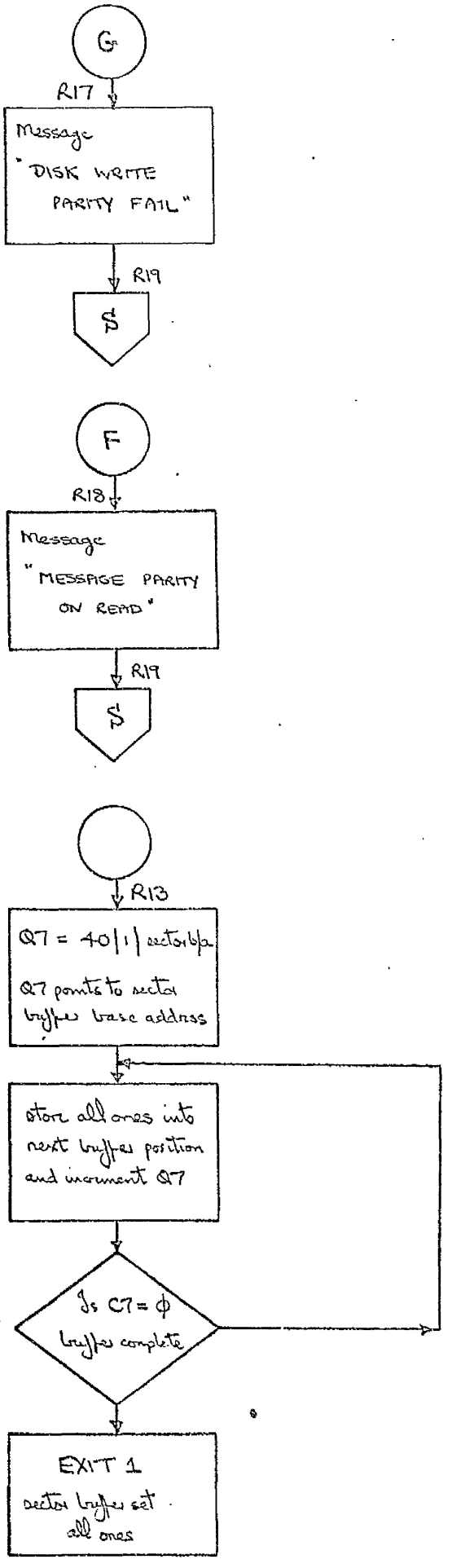

C
Is C4 = ϕ sector buffer full
no
R2
N
O
yes
R21
Place 6 dummy BCD teletype characters in N2
R24
Is C5 = ϕ N1 left justified
yes
no
Move one dummy character from N2 into D40-47 of N1 DC5
R25
Store left justified word into next position in sector buffer
R7
Copy and increment next sector address in V3
Is C15 = ϕ message finished
no
R8
P
yes
D

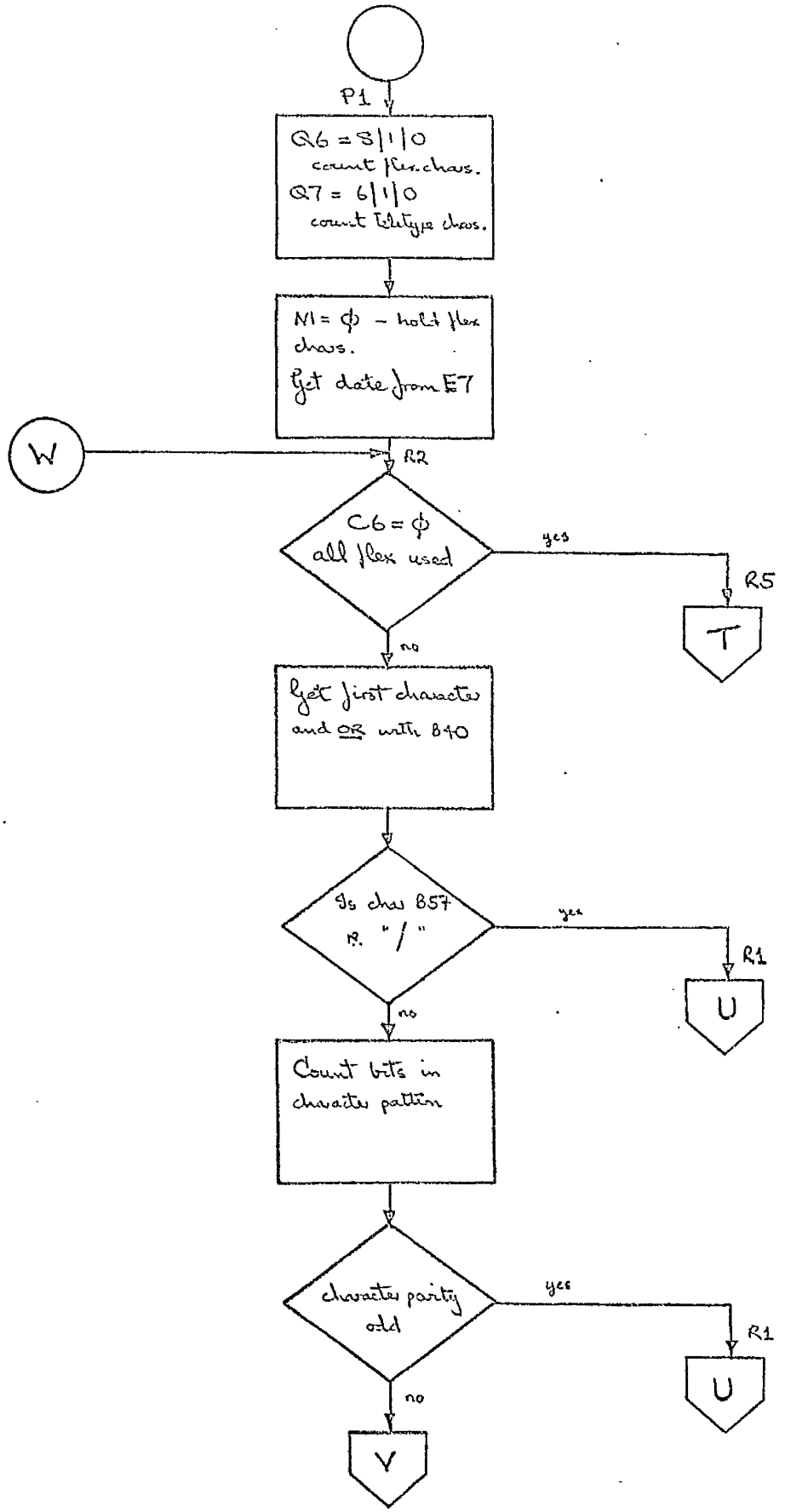

D
R9
Find number of words stored in sector; OR with N2 = ϕ or NOC address
Store into Y29+, no. 838 of sector bytes - the link word
R10
OUT4-1
write sector to disk reset Q+ to sector bytes b/a
OUT4-7
check parity on last disk transfer
disk transfer parity
yes
R17
G
no
JS13
clear disk buffer to all ones
Set C15 = -1 ready to process next message
R6
E

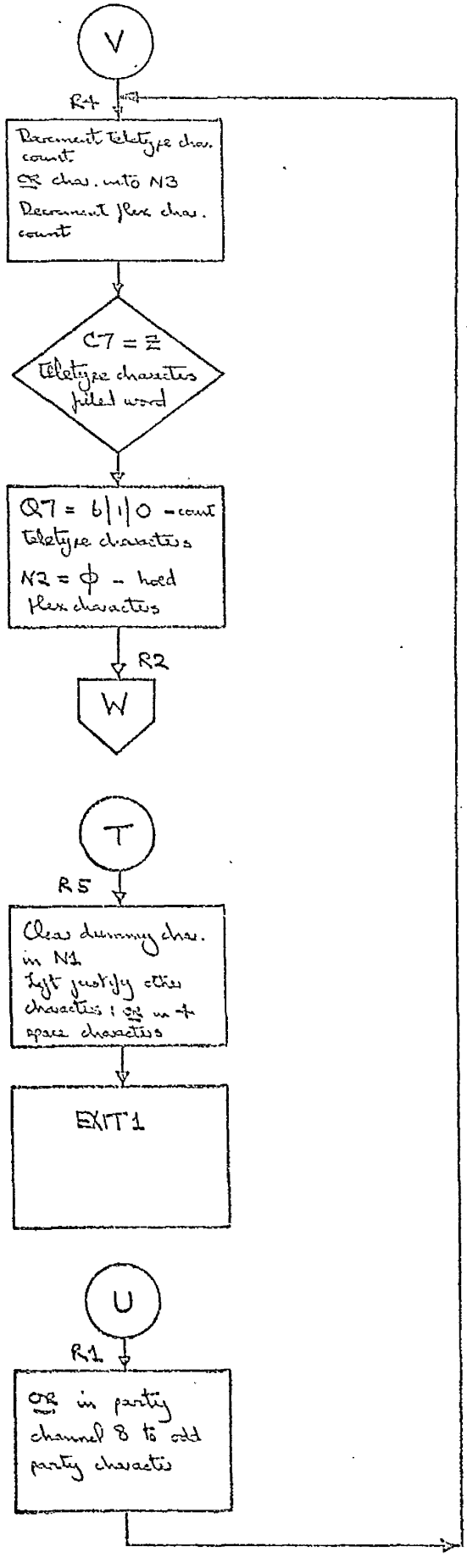

R16
PIDQ1
character read to end message on PTR
Test parity on PTR
parity ?
yes
R18
F
no
Q2 = 228 | 1 | buffer b/a
Q3 = 6 | 1 | 0
character pack counter
EXIT 1
M
R11
C14 = 0
mark all messages as processed
L
R4
Save Q3 in Q5
Clear Q2, Q3
Clear C15 - end of block ; replace ← by dummy 8377
R5
Q


P
R8
Get V3, Next sector address in M part of N1
R9
D
K
R15
Clear character in N1
Is C2 = ϕ PTR buffer empty
no
R22
R
J
yes
R20
JS16
PTR character read
R22
R


H
R12
Deallocate PTR
JSP1
get date in teletype chars.
JS3L2
clear core and Q stores
Set N1 and N2 to identifier of CommCon
OUT 1
Call in CommCon
S
R9
OUT 8
Output failure message
OUT ϕ
Program failure
PROGRAM ENDS


G
R17
Message
"DISK WRITE
PARITY FAIL"
R19
S
F
R18
Message
"MESSAGE PARITY
ON READ"
R19
S
R13
Q7 = 40|1| sector bfr
Q7 points to sector
buffer base address
store all ones into
next buffer position
and increment Q7
Is C7 = ϕ
buffer complete
EXIT 1
sector buffer set
all ones


P1
Q6 = 8|1|0
count flex chars.
Q7 = 6|1|0
count teletype chars.
N1 = ϕ - hold flex chars.
Get date from E7
W
R2
C6 = ϕ
all flex used
yes
R5
T
no
Get first character and OR with 040
Is char B5? = "/"
yes
R1
U
no
Count bits in character pattern
character parity odd
yes
R1
U
no
V


V
R4
Decrement teletype char. count
≠ char. into N3
Decrement flex char. count
C7 = 2
Teletype characters filled word
Q7 = 6|1|0 - count teletype characters
N2 = ϕ - load flex characters
R2
W
T
R5
Clear dummy char. in N1
Left justify other characters ; ≠ in + space characters
EXIT 1
U
R1
OR in parity channel 8 to odd parity characters

Coding and Flow charts of Communication Controller

ComDon

```
ST 2000;
TL 9999;
RESTART; J3; J3;
PROGRAM;
      (ENTER WITH N1,N2 = DATE IN TELETYPE CHARS;
            N3 = ICM OUTPUT DEVICE NUMBER; N4 = ICM
            INPUT DEVICE NUMBER);
      TO; =V39P899; =V40P899; (DATE);
      DUP; V36P899; OR; =V36P899;
      V38P899; OR; =V38P899;
            (OUTPUT PARAMETERS);
      DUP; V35P899; OR; =V35P899;
      V37P899; OR; =V37P899;
            (INPUT PARAMETERS);
2;      V21P899; =RP1; (SET UP B/A OF FIFO TABLE);
      Q1TOQ4; (PERMANENT SETTING);
      V43P899; =RC2; (NO. OF TERMINALS);
4;      SET-256; =MON1Q;
      ZERO; NOT; DUP; =MON1Q; =MON1Q;
      DC2;
      J4C2NZ; (RESET FIFO);
      SET AV0P896; SET -9; OUT; (IPCB B/A);
      V38P899; JSP905; (SET UP FIRST BUFFER);
1;      V35P899; V37P899; DUP; =Q14;
      =V35P899; =V37P899; (INPUT BUFFER SWAP);
      V36P899; V38P899; DUP; =Q15;
      =V36P899; =V38P899; (OUTPUT BUFFER SWAP);
8;      JSP803; J8; (TINT B1/2/3);
      PWQ15; PRQ14; (SET UP TRANSFERS);
      V37P899; JSP800; (DEAL WITH INPUT BUFFER);
      V38P899; JSP905; (SET UP OUTPUT BUFFER TO
            ERASE CHARACTERS AND TERMINAL ADDRESSES);
      V35P899; JSP900; J1; (CHECK LOCK-OUT);
      V2P899; V3P899; OR; J5=Z;
      JSP821; (DEAL WITH OUTSTANDING COMMANDS);
      V35P899; JSP900; J1; (EXIT 1 IF TRANSFER FINISHED);
5;      V34P899; NOT; J1=Z;
      JSP808; (OUTPUT);
      J1;
3;      J10EJ; LINK; ERASE; J3;
10;      J11EN; ERASE; J10;
11;      J2;
```

```
P900; (CHECK LOCK-OUT);
      ZERO; =TR;
      =Q1; TLOQ1;
      J1TR; EXIT 1; (TRANSFER ENDED);
1;      EXIT 2; (TRANSFER STILL IN PROGRESS);


P905; (RESET OUTPUT BUFFER);
      SET 10; =RC15;
      SHL-16; =M15;
      V41P899;
£1;      DUP; =NON15Q; £J1C1SNZS;
      ERASE; EXIT 1;


P906; (SET N9 TO SECTOR BA FOR DESK GIVEN IN N1);
      SET 40; *D; CONT;
      V15P899;
      +; =N9;
      EXIT 1;


P910; (COMMAND SUSPENSION);
      ZERO; =TR;
      M9TOQ1; N1; =11; SET 39; =+11; (Q1 = BUFFER LIMITS);
      TLOQ1; J1TR; (LOCKED OUT);
      J1P804; (CLEAR SUSPENSION MARKER IN V3P899 AND EXIT);
1;      JS2P804; (SET SUSPENSION MARKER);
      LINK; ERASE; (REMOVE LINK TO P910);
      EXIT 2; (SUSPEND COMMAND FOR THIS TERMINAL);


P911; (SET M7 TO OCW1 FOR TERMINAL IN V23P899);
      V23P899; DUP; SHL+1; (LEAVES TERMINAL ADDRESS IN N1);
      SET AVOP809; =RM7; =+M7;
      EXIT 1;


P912; (SET M7 TO FIFO FOR TERMINAL IN V23P899);
      V23P899; DUP; SHL+1; +;
      V21P899; =RM7; =+M7;
      EXIT 1;


P913; (GET FIFO AND CLEAR);
      NOM7; SET -256; =NON7Q;
```

```
SHL-8; SHL+8; (CLEAR TERMINATOR);
MOM7; ZERO; NOT; DUP; =!MOM7Q;
MOM7; REV; =MOM7;
EXIT 1;
```

```
P800;(DEAL WITH INPUT BUFFER);
      SET 10; =RC3;
      SHL-16; =N3; (SET UP BUFFER COUNTER);
      ZERO; (DUMMY IN N1);
1;      J10C3Z; ERASE;
      (EXIT IF WORD COUNT ZERO);
      SET 4; =RC2; (GROUP COUNT);
      MON3Q; (FETCH BUFFER WORD);
2;      DC2;
      ZERO; SHLD+12; SHC-8;ZERO; SHLD+8; (LEAST SIG. 8 BITS IN N1);
      SET B377;
      J4*NE; ERASE; ERASE; J2C2NZ; (£ERASE£ CHAR. FOUND);
      J1;
4;      SET B300; J5=;
      SET B243; J6=; (DELETE FIFO NEW WORD);
      REV; SHL-8;
      JSP810; (STORE CHAR. IN N2 INTO FIFO
            OF TERMINAL IN N1);
7;      J2C2NZ; J1;
5;            (FIND COMMAND TERMINATOR);
      REV; SHL-8;
      JSP807; (SET FLAG);
      J7;
10;      ERASE; EXIT 1;
6;      ERASE; SHL-8;
      JSP811; (RETYPE OLD FIFO);
      J7;
```

```
P80;V30; (GET NEXT SECTOR ADDRESS FOR GIVEN
          TERMINAL; IF SECTOR ADDRESS *GT 15 GET NEW BLOCK AND
          UPDATE ASL; 15 WORDS OF CURRENT BLOCK/SECTOR ADDS.;
          16 WORDS OF ASL. 8 FOR EACH DISC);
      V8=B 7770 0000 0000 0000; (ASL PRESET);
      SET AV0; =RN12; (SET UP ADD. LIST BA);
      =N13; (DESK ADDRESS);
      N12N13; (FETCH CURRENT DISK ADD. FOR THIS DESK);
      DUP; SHC+12; ZERO; SHLD*4;
      DUP; SET 16; SIGN;
      J1*GEZ; (J IF BLOCK FILLED);
      NOT; NEG;
      SHLD-4; ERASE;
      SHC-12;
      =N12N13; (RESTORE NOC ADDRESS);
      EXIT 1; (NEXT SECTOR IN OLD BLOCK IN N1);
1;      ERASE; ERASE; ERASE; (SEARCH ASL FOR NEXT FREE BLOCK);
      JS6; DUP; NOT; NEG; SHLD+36; =N12N13;
      EXIT 1;
6;      SET AV15; =RN14; SET 16; =RC15; (SCAN ASL);
4;      N14N15Q; DUP; NOT; J5=Z; SET 48; =C14;
£3;      DUP; J2*GEZ; SHL+1; DC14; J3C14NZS;
5;      ERASE; J4C15NZ;
      SET B24; JP829; (CATASTROPHIC FAILURE; NO DISK SPACE);
 2;      (FIND FREE BLOCK);
      ERASE;
      M-115; N15; (ADDRESS OF CURRENT WORD);
      SET 48; *D; CONT; (GET CORRES. BLOCK ADDRESS);
      SET48; C14; -; (ADD. OF CURRENT BIT);
      DUP; =C14;
      N14N15;
      SHCC14; ZERO; NOT; CONT;
      NC14; SHCC14; =N14N15; (SET ASL BIT ON);
      +; SHL+4; (*16 TO ALLOW FOR SECTOR ADDRESSES);
      EXIT 1;
```

```
P802; (STORE FULL FIFO DUMP WORD INTO DISK BUFFER);
      M2; DUP; JSP906;
      SET AVOP898; =M11; =+N11;
      MOM11; (SECTOR PACKING CONTROL WORD);
      =Q10;
      #M9M10Q; (STORE FIFO DUMP WORD);
      JIC10Z;
      Q10; =MOM11;
      EXIT 1;
1;      VOP899; =MOM11; (RESET);
      M2; JS2P806; (ACCESS DFW2);
      DUP; =Q11;
      SET 30; C10; -; (FOR PARTIAL USE BY P828);
      JS2;
      SHL+16;
      M2; JSP801; SHL-32; OR;
      (GET NEXT SECTOR ADDRESS AND OR INTO
            C-PART OF DFW2);
      =MOM8N; (NEW DFW2);
3;      SET 47; OUT; ZERO; =TR;
      Q11;
      SET 41; OUT; (WRITE UP SECTOR);
      M2; JSP806;
      SHC-32; NOT; NEG; SHC+32; (INCREMENT
            SECTOR COUNT IN L.S. END OF C-PART
            OF DFW1);
      =MOM8;
      EXIT 1;
2;      =I11; (NUMBER OF WORDS OF DATA);
      Q11; =E38M9;
      ZERO; NOT; =E39M9;
      DUP; SHL-16; =C11; (CURRENT DISK
            ADDRESS FOR THIS BUFFER);
      M9; DUP; =I11; =M11;
      SET 39; =+N11; (Q11 IS OUT 41 WORD);
      EXIT 1;
```

```
P803V6; (INSPECT TINT B);
        V0/2=P *Q8S*U GOLD£SYS£MODE;
        V3/4=P DATALINK GOLD£SYS;
        V5=Q 0/AV0/AV2;
        V6=0; (ZERO IF GOLD OPERATIONAL);
        MOMO; MOMOH;
        DUP; PERM;
        NEV;
        DUP; J1*GTZ;
2;        ERASE; ERASE;
        J9;
7;        EXIT 2;
1;        DUP; SET 3; SIGN; J8*GTZ;
        NEG; NOT;
        =LINK;
        =MOMO;
        EXIT AR3;
£3;        J4; (TINT B1 = TO DATALINK);
        £J5; (TINT B2 = TO RJE SYSTEM);
£6;        (STATUS ENQUIRY);
        ZERO; =V0;
        V5; SET 8; OUT;
9;        V6; J7=Z;
        EXIT 1;
8;        ERASE; =MOMO;
        J9;
4;        (SWITCH TO DATA LINK);
        V6; J6*NEZ;
        ZERO; NOT; =V6; (MARKER);
        V3; =V1;
        V37P899; SHL-32; SET 6; OUT; (INPUT);
        V36P899; SHL-32; SET 6; OUT; (OUTPUT);
        ZERO; =V0;
        V5; SET 8; OUT;
        EXIT 1;
5;        (RESUME ONLINE SYSTEM);
        V6; J6=Z;
        ZERO; =V6; (RESET MARKER);
        V4; =V1;
        SET B65; SET 5; OUT;
        ERASE;
        SET B66; SET 5; OUT;
        ERASE;
        ZERO; =V0;
        V5; SET 8; OUT;
        EXIT 2;
```

```
P804; (ALTER SUSPENSION MARKER IN V3P899; TERMINAL
          ADDRESS IN C7);
1;      V3P899; SHCC7; STR; ERASE; J3; (CLEAR);
2;      V3P899; SHCC7; ZERO; NOT; CONT; (SET);
3;      NC7; SHCC7; =V3P899; EXIT 1;


P805V3; (COMPOSE REAL TIME);
      V0=B 1212 1212 1212 1212;
      V1=B 6060 6060 6060 6060;
      V2=0; (STORE TIME DURING ASSEMBLY);
      V3=0; (MARKER);
      SET 9; OUT; SHL-24;
      SET 3600; *DIV; (MINS/HOURS);
      SET 60; *DIV; ERASE;
4;      V0; REV; FRB;
      V1; OR; SHL+36; (MS 12 BITS ARE MINS);
      SHL-36;
      JS3;
      V3; J5*NEZ;
      ZERO; NOT; =V3;
      SET B56; SHL+40;
      V2; SHL-8; OR; =V2;
      J4;
3;      SHC-6;
      ZERO; SHLD*6;
      DUP; BITS; SHC-1;
      J1*LTZ;
2;      SHL+40;
      V2; SHL-8; OR; =V2; SHL-6;
      DUP; J3*GTZ;
      ERASE;
      EXIT 1;
1;      SET B200; OR; J2;
5;      SET B240; SHL+40; V2; SHL-8; OR;
      ZERO; =V2; ZERO; =V3;
      EXIT 1; (TIME IN N1);
```

```
P806;(ACCESS DESK FLAG WORD; ADD.OF DESK IN N1);
1;      SHL+1; V14P899;
       +; =RM8;
       MOM8;
       EXIT 1;
2;      JS1; ERASE;
       MOM8N; (DFW2);
       EXIT 1;


P807; (D2 OF DFW1 SET IF COMMAND TERMINATOR;
            CHANGE STATUS IN V34P899);
       (ADD./CHAR./WORD);
       DUP;JSP806; (ACCESS DESK FLAG WORD);
       SHC+2; ZERO; NOT; CONT; SHC-2;
       =MOM8;
       JSP812; (ALTER I/O STATUS);
       JSP810;
       ZERO; NOT; =V2P899; (FORCE COMMAND SEARCH);
       EXIT 1;


P810;(STORE CHAR; ADD./CHAR);
       DUP; =N2; (RECORD DESK ADD.);
       DUP; SHL+1; +;
       =RM5;
       M4M5Q; M4M5Q; DUP; M4M5;
       DUP; J1*GEZ; (FIFO FULL);
       SHLD+8;
2;      I5=-1; =M4M5Q;
       ERASE; SHLD+8;
       =N4M5Q; OR; =N4M5; EXIT 1;
1;      SHLD+8; JSP802; (STORE DUMP WORD);
       SET -256; J2; (NEW DUMP WORD);
```

```
P811; (RETYPE OLD FIFO; DELETE NEW);
      DUP; JSP806; SHL+3; (DFW1);
      J3*LTZ; (J IF PROMPT RESPONSE DUE);
      JSP812;
      DUP; SHL+1;
      DUP; PERM; +; =RM7;
      SET -256; =M4M7Q; (CLEAR NEW);
      M4M7Q; M4M7;
      SET 3; =C7; (NO OF WORDS TO BE TYPED);
      CAB; =M7;
      V1P899;
1;      SET AVOP809; =M5; (B/A OF OCW TABLE);
      V15P899; =RM15; (B/A OF DATA BUFFERS);
      M7; SET 20; *D; CONT;
      SET 30; +;
            (TERMINAL DATA=SECTOR B/A + 30; THE GAP);
      DUP; =RN14;
      +; =M5M7; (UPDATE CURRENT WORD POINTER FOR OCW1);
      ZERO; NOT; SHL-16; =M5M7N; (OCW2 = 0/-1/-1);
      V20P899;
£2;     DC7; =M14M15Q; £J2C7NZS; (STORE WORDS FROM NEST);
      EXIT 1;
3;      JSP906; (PROMPT RESPONSE DUE; SET M9 TO DATA SECTOR B/A);
      E22M9; J4*NEZ; (J IF RESET INHIBITED);
      ZERO; =E11M9; (RESET PARAMETER COMMAND);
4;      EXIT 1;


P812; (ALTER I/O STATUS);
      DUP; =C8;
      V34P899; SHCC8;
      STR; NOT; CONT;
      NC8; SHCC8;
      =V34P899;
      EXIT 1;


P813; (BINARY TO TELETYPE);
      V0P805;
      REV; FR8;
      V1P805; OR;
      SHL+12; SHL-12; (ONLY PACK 6*8 BIT CHARS);
      JS3P805;
      V2P805; (CODE CONVERTED NUMBER);
      ZERO; =V2P805;
      EXIT 1;
```

```
P814V0; (TELETYPE TO BINARY);
6;      (CHECK TIME);
      JS5; (GET BINARY NO. FROM FIFO IN N1,N2,N3);
      V0; DUP; SET 7200;
9;      VR; SIGN; J7*GTZ;
      J7TR; (NON-NUMERAL FOUND);
      EXIT 2;
8;      (CHECK STORE);
      JS5;
      V0; DUP; SET 14100;
      J9;
5;      ZERO; =TR;
      ZERO; =V0; SET 3; =C15; (COUNT FIFO);
3;      SET 6; =C14;
2;      ZERO; SHLD+8; DC14;
      SET B377; J1=;
      DUP; J1=Z;
      SET B177; AND;
            (CHECK FOR NON-NUMERIC);
      DUP; SET B60; SIGN; J10*LTZ;
      DUP; SET B71; SIGN; J10*GTZ;
      SET B17; AND;
11;     (SCALE AND ADD DIGIT);
      V0; SET 10; *D; CONT;
      +; =V0;
4;      J2C14NZ; DC15; ERASE;
      J3C15NZ;
7;      EXIT 1;
1;      ERASE; J4;
10;     ZERO; NOT; =TR; (FAIL MARKER);
      J11;
12;     JS5; V0; DUP; SET 3072;
      J9; (CHECK DISC ADDRESS);
```

```
P821; (SEARCH FOR COMMAND MARKER D2 IN DFW1 OF DFW TABLE);
      V14P899; =R5;
      V43P899; =C5; I5=+2;
1;      M5TQQ3; MQM5Q; (GET DFW1);
      SHL+2; J2*LTZ;
10;       J1C5NZ; ZERO; =V2P899;
      EXIT 1; (FINISHED ALL COMMANDS);
2;              (DETECT COMMAND MARKER);
      V43P899; C5;
      -; NEG; NOT;
      DUP; =C8;
      =V23P899; (ISSUING TERMINAL);
      MQM8; (CURRENT DFW1);
      V3P899; SHLC8; J20*LTZ; (J IF COMMAND SUSPENDED);
      DUP;
      SHL+3; J11*LTZ;
            (JUMP IF PROMPT RESPONSE DUE);
      ERASE; Q5;
      (SAVE Q5 IN N5);
      JSP822; (EXTRACT COMMAND NAME FROM FIFO);
      J5; (EXIT 1 P822 UNABLE TO EXTRACT NAME
      N1=Q5 IF FIFO SEARCHED TO COMPLETION);
      JSP823; (EXIT 2 P822 WITH VALID NAME; USE NAME
            IN TABLE LOOK-AT);
      J6; (EXIT 1 P823 INVALID NAME);
            (3 ITEMS IN NEST; Q5 IN N4);
      J7; (EXIT 2 P823 - SPECIAL TERMINATOR; 4 ITEMS IN
            NEST; Q5 IN N5);
            (EXIT 3 P823 COMMAND PARAMETER WORD IN N1; Q5 IN N2);
      J8TR; (JUMP IF IN INTERROGATIVE MODE);
      DUP; SHL+1; J4*GEZ; (SAVE DATA);
      V23P899; DUP; =N1;
      SET AVOP898; =+N1;
      SET 30; MOM1; SHL-32; -;
            (NO. OF WORDS IN BUFFER);
      REV; JS2P806; SHL-16; SHL+16; (CLEAR TRAILER);
      DUP; =Q11; REV; JS2P802;
      ERASE; (CLEAR COPY OF DFW2; Q11 IS TRANSFER PARAMETERS);
      C11; JSP831;
      V23P899; =N2; (FOR P802);
      JS3P802; (WRITE UP SECTOR TO DISK);
4;      JSP824; (ENTER COMMAND GIVEN IN N1);
      J16; (EXIT 1 - COMMAND COMPLETED);
      J18; (EXIT 2 - COMMAND SUSPENDED);
16;       =Q5; (RESTORE Q5);
      J3; (CONTINUE CYCLE FOR OTHER DESKS);
5;              (UNABLE TO FIND VALID COMMAND FORMAT IN FIFO);
      =Q5;
      V6P899; V5P899; V4P899;
      JS11P841; J3;
6;              (COMMAND NOT IN DICTIONARY);
      ERASE; ERASE; ERASE;
```

```
        J5;
7;                  (COMMAND HAS SPECIAL TERMINATOR; UNUSED IN RJE SYSTEM
        JSP827; J3; (DEAL WITH SPECIAL ACTION; RESTORE Q5
              BEFORE LEAVING P827 DUMMY ROUTINE);
8;        JSP828; (TYPE DESCRIPTIVE MESSAGE; RESTORE Q5 IN P828);
        J3;
3;        (CLEAR DFW1 D2);
        V23P899; JSP806;
        SHC+2; STR; ERASE; SHC-2;
        =MON8;
        J10;
11;                 (END OF PROMPT RESPONSE);
        Q5; REV; (DFW1 IN N1);
        SHL+6; SHL-46; DUP; =LINK;
        SHC-1; J13*LTZ;
        EXIT AR14;
13;       EXIT 1 AR14;
£14;       J17; J15;
        J17; J15;
15;       JS1P843; (LIST COMMAND);
        J16; (EXIT 1 - COMPLETED);
        J18; (EXIT 2 - SUSPENDED);
17;       JS1P841; (CALL + PROMPT COMMANDS);
        J16; (EXIT 1 - COMPLETED);
        J18; (EXIT 2 - SUSPENDED);
18;       =Q5; J10; (COMMAND SUSPENDED FOR THIS TERMINAL
              DOES NOT CLEAR D2/3 OF DFW1; SET V3P899 TO
              FORCE RE-ENTRY TO P821);
20;                 (RE-ENTER SUSPENDED COMMAND);
        SHL+6; SHL-30; (PUT COMMAND NO. IN I-PART);
        Q5; REV; J4; (ENTER COMMAND AGAIN VIA P824);
```

```
P822; (EXTRACT COMMAND NAME FROM FIFO; SET TR
        IF INTERROGATIVE MODE; STORE OTHER CHARS.
        INTO DATA-BUFFER);
      ZERO; DUP; =V25P899; =V26P899; (CLEAR NAME STORES BEFORE USE).
      V21P899; =RN5; (BA OF FIFO);
      V23P899; (SET UP FETCH OF FIFO);
      DUP; SHL+1; +; =+N5;
      MOM5; SET-256; =MOM5Q;
      MOM5; ZERO; NOT; =MOM5Q;
      MOM5; ZERO; NOT; =MOM5; (FETCH FIFO WORDS AND DUMMY FIFO ITSEL
      SET 3; =RC7; (COUNTS WORDS OF FIFO);
3;      SET 6; =RC5; (COUNT CHARS. IN FIFO WORD);
2;      ZERO; SHLD+8; (MOVE TOP CHARS INTO N1);
      DC5;
      SET B134; (CHECK FOR COMMAND WARNING CHAR., CWC);
      J4=; (J IF CWC FOUND);
      V24P899; SHL+8; OR;  (V24 ACTS AS DUMMY
        FIFO DUMP WORD);
      J1C5NZ; Q7; REV; (SAVE Q7 IN N5);
      V23P899; =M2; JSP802; (COPY WORD TO BUFFER AS FULL,);
      =Q7; DC7;
      SET-256; =V24P899;
        (RESET DUMMY DUMP WORD);
      ERASE; (CLEAR FIFO WORD TO ACCESS NEXT ONE);
      J3C7NZ; (PROCESS NEXT WORD);
      EXIT 1; (FAIL TO FIND CWC; NEST EMPTY
        AFTER GOING ROUND LOOP 3 TIMES);
1;      =V24P899; J2;
4;      ERASE; (DELETE CWC);
6;      SET 6; =RC8; (COUNT CHARS. INTO V26,V25P899);
      J13C5Z;
7;      ZERO; SHLD+8; DC5; (DECREASE FIFO CHAR COUNT);
      SET B300; J9=; (J IF COMMAND TERMINATOR CHAR., CTC, FOUND);
      V26P899;
      SHL+8; OR; DC8; (MOVE UP ONE CHAR. POSITION);
      J5C8NZ; (JUMP IF V26 NOT FILLED);
      V25P899; SHLD+8;
      =V25P899; SHL-8; (SPILL ONE CHAR INTO V25);
      =V26P899; SET 1; =C8; J8; (SET C8 = 0 AFTER NEXT
        ENTRY AT R7);
5;      =V26P899;
8;      J7C5NZ; (JUMP IF FIFO WORD NOT EMPTY);
13;      ERASE; SET 6; =RC5; DC7; J7C7NZ;
      EXIT 1; (FAIL TO FIND CTC);
9;      ERASE; SET B377; SHLD-8; (CLEAR TERMINATOR);
      ERASE; (CLEAR ZERO LEFT BY SHLD-8);
      V26P899; SHL+8; V25P899;
       SHLD-8; (NAME RIGHT JUSTIFIED IN N1,N2);
      =V25P899; =V26P899; (NAME STORED);
      JS10; J11C5NZ; (CHARS. LEFT AFTER CTC);
      DC7; ERASE;
      J12C7NZ;
```

```
        EXIT 2; (DUMMY DUMP WORD COPIED TO DATA SECTOR
              AND EXIT 2 IF ALL FIFO PROCESSED);
11;        (DUMP REST OF CHARS);
        JS2; JS10;
        EXIT 2;
12;        (DUMP REST OF CHARS);
        JS3; JS10;
        EXIT 2;
10;        Q7;V24P899; V23P899;
        =M2; (PREPARE TO STORE DUMMY DUMP WORD INTO
              DATA SECTOR);
        JSP802; =Q7; EXIT 1;
```

```
P808; (OUTPUT CONTROL SUBROUTINE FOR 4 DESKS);
      V15P899; =Q15; (BA OF 640 WORD OUTPUT BUFFER);
      V43P899; =RC7; (NO. OF DESKS);
      SET AVOP809; =RM5; (B/A OF OCW TABLE);
      V34P899; (I/O STATUS WORD);
£2;     DUP;
      J1*GEZ; (JUMP IF DESK IS RECEIVING);
10;     SHL+1; (SHIFT UP I/O STATUS WORD TO NEXT DESK);
      £DC7; M+I7; (UPDATE);
      J2C7NZS; (JUMP IF STILL DESKS TO DEAL WITH);
      ERASE; (CLEAR V34P899);
      EXIT 1;
1;      (SELECT BUFFER);
      SET 10; =RC9;
      V38P899; SHL-16; =M9; (OUTPUT BUFFER PARAMETERS);
      M7; SHL+1; (*2);
      =RM8; M5M8; (WORDS LEFT/CHARS LEFT/CURRENT WORD);
30;     =Q13; C13; J12*LTZ; (NO OUTPUT AVAILABLE YET);
      I13; =RC14; (Q14=CHARS LEFT/1/0);
      I13=+1;
      J6C13Z; (J TO LOAD NEW SECTOR ; USE OCW2);
      V43P899; M7; -; NEG; NOT;
      SHL+2; DUP; SHL+1; +; =C11; (DISTANCE TO MOVE CHAR FOR OUTPUT);
      (Q13=WORDS LEFT/1/ADD OF CURRENT WORD);
7;      M15; M13; +;
      DUP; SET 2; +;
      =M11; =I11;
      TLOQ11; J10TR; (LOCKOUT ON - AVOID LOV);
      M15M13;
5;      ZERO; NOT;
      SHLD+8; DC14; (GET NEXT OUTPUT CHAR.);
      SHCC11; MOM9; AND; =MOM9Q; (AND CHAR INTO OUTPUT WORD);
            (ONLY 4 TERMINALS);
      J3C14Z; (JUMP IF OUTPUT WORD EMPTY);
      J4C9Z; (JUMP IF ALL OUTPUT BUFFER WORDS DEALT WITH);
      J5;
3;      ERASE; (CLEAR EMPTY OUTPUT WORD);
      DC13; M+I13; (UPDATE NEXT WORD FETCH);
      J6C13Z; (JUMP IF 40 WORD OUTPUT SECTOR CLEARED);
      SET 6; =RC14;
      J13C9Z; ( J IF ALL OUTPUT BUFFER WORDS DEALT WITH);
      J7;
6;      M5M8N;
      =Q11;  (OCW2);
      J8C11NZ; (J IF SYSTEM MESSAGE READY);
      M11; J26=Z; (NOC ADDRESS);
      I11; DUP; J20*LEZ; (WAIT FOR REST OF OUTPUT);
      ERASE;
31;     M11; DUP; J9=Z; (J IF NO MORE OUTPUT);
11;     SHL+32; (MOVE DISK SECTOR ADDRESS);
      M7; SET 40; *D; CONT;
      DUP; DUP; =M13; (RESET BA);
```

```
        M15; +; DUP; SHL+16;
        REV; SET 39; +; OR;
        CAB; OR; (SET UP READ; N2=SECTOR BA RELATIVE TO M15);
              (3 ITEMS IN NEST; READ PARAMS/SECTOR
              BASE/V34P899);
        ZERO; =TR;
        DUP; =Q1; TLOQ1; J70TR;
        Q8; M7; JSP806; (DFW1 IN M1);
        DUP; J71*LTZ; (J IF TRANSFER HAS BEEN STARTED);
              (NEST = DFW1/Q8/READ/B-A/PATTERN);
        ZERO; NOT; CONT; =MOM8; (SET DO IN DFW1 = TRANSFER STARTED);
        =Q8; (RESTORE);
        SET 47; OUT; ZERO; =TR; (PARITY IGNORE);
        SET 42; OUT; (READ DOWN NEXT SECTOR);
        ERASE; (CLEAR SECTOR B/A REL TO M15);
        J10; (SUSPEND ON BUFFER LOCKOUT);
71;           (TRANSFER NOW COMPLETE);
        STR; ERASE; (CLEAR DFW1 DO = TRANSFER STARTED MARKER);
        =MOM8; =Q8; (RESTORE);
        ERASE; I11; NEG; NOT; =I11; (B-A/PATTERN);
              (DECREMENT COUNT OF AVAILABLE SECTORS);
        SET 38; =+M13; (SECOND LAST WORD IS SECTOR LINK);
        M15M13; (PREVIOUS SECTOR LINK/NO. DATA WORDS/ FOLLOWING
              SECTOR LINK);
        SHC-16; ZERO; SHLD+16;
        DUP; SET 3072; SIGN; J29*GTZ;
        =M11; Q11; =M5M8N;
              (UPDATE NEXT SECTOR POINTER IN OCW2 M-PART);
        SHL+16; SHL-16;
        DUP; SHL-16; SET 38; SIGN; J27*GTZ;
28;        SET 6; OR; SHC+16; OR;
        DUP; =M5M8; (UPDATE WORDS LEFT/ CHARS. LEFT/CURRENT
              WORD POINTER);
        J30;
29;        ERASE;
27;        ERASE; ERASE;
        V9P899; V8P899; V7P899;
        JS11P841; (OUTPUT FUNNY LINK WORD MESSAGE);
        J10;
4;        =M15M13; (RESTORE OUTPUT WORD);
13;        C14; =I13; (RECONSTRUCT OCW1);
        Q13;
        =M5M8; (SAVE OCW1);
        J10;
9;        ERASE; (M11);
26;        ZERO; NOT; =M5M8; (SET C PART -VE; NO AVAILABLE OUTPUT);
        ZERO; =M5M8N; (CLEAR OCW2 = FILE OUTPUT COMPLETE);
12;        Q8; M7; JSP812;
        ERASE; =Q8;
        J10;
8;        C11; COTOQ11; Q11; (SYSTEM MESSAGE READY = CLEAR MARKER);
        =M5M8N; J11; (CHAIN ON SYSTEM MESSAGE);
```

```
20;     J26*LTZ; (NO MORE OUTPUT DUE IN THIS SEQUENCE);
      SET AVOP896; =RM12; (GOAW LIST BA);
      M8; SHL-1; =M13; (DESK ADD);
      M12M13; (GET GOAW);
      J26*LTZ; (OUTPUT FINISHED OR OFFLINE OUTPUT SELECTED);
      ZERO; =TR;
      SET AV16P896; =RM3;
      M3M13; (DOAW);
      SHL+32; DUP; SHL-32; DUP;
      CAB; J23*LTZ; (FINAL OUTPUT AVAILABLE);
24;     . M12M13; (GOAW);
      DUP; SHL+16; SHL-32; (NO OF SECTORS OUTPUT SO FAR);
      REV; PERM; -; (NO. SECTORS AVAILABLE FOR OUTPUT);
      DUP; J21=Z; (NO FURTHER SECTORS AVAILABLE YET);
      MSM8N; =Q11; =I11; (COPY OF NEW OCW2);
      (SET UP NEW NO OF SECTORS AVAILABLE FOR THIS SEQUENCE);
      J25TR; SHL+32; OR; SHC+16;
      =M12M13; (GOAW);
      J31;
23;     ERASE; SHL+32; SHA-32; (FINAL
          OUTPUT AVAILABLE);
      NEG; DUP; (ABS. VALUE OF DOAW);
      ZERO; NOT; =TR; (SET END OF INPUT FLAG);
      J24;
25;     ERASE; ERASE;
      ZERO; NOT; =M12M13; (OUTPUT COMPLETE);
      J31;
21;     ERASE;
70;     ERASE; ERASE; J10; (CONTINUE TO NEXT
          TERMINAL);
```

```
P809V15; (OUTPUT CONTROL WORDS; 2 PER DESK);
       V0=-1; (OCW1 FOR DESK 0);
       V2=-1;
       V4=-1;
       V6=-1;
       V8=-1;
       V10=-1;
       V12=-1;
       V14=-1;

P831; (OR POINTER IN N1 INTO DFW1);
       NOM8; SHL-16; SHL+16; (CLEAR BITS FROM TRAILER POINTER);
2;       OR;
       =NOM8; EXIT 1;
1;       NOM8; SHC+16; SHL+16; SHL-16; SHC-16; J2;
              (CLEAR LEADER POINTER);
```

```
P823: (TABLELOOK-AT USING 2 WORD COMMAND NAME);
      V25P899; ZERO; V26P899; (NAME IN N1(L.S.) AND
            N3(M.S.));
      SHLD-8; (PUT TERMINATOR INTO N2);
      REV; SHL-40;
      SET B56; (TEST FOR .IN TELETYPE CODE);
      J4=;
      DUP; SET B77; AND; (CLEAR OUT M.S.TWO BITS);
      DUP; SET 39; SIGN; (TEST IF CHAR.VALUE *GT39);
      J1*LTZ; (J IF VALUE *LT 39);
2;      ERASE; (ERASE ALTERED CHAR.IN N1);
      ZERO; NOT; =TR; (SET TR TO INDICATE TEST-MODE);
      PERM; SHL+8;
      REV; SHLD-8;
      CAB; (LAST CHAR.IN N1.,REST OF NAME R.J. IN N2/3);
      J5;
1;      DUP; SET 33; -; DUP;
      J3*GEZ; ERASE; J2;
3;      REV; ERASE; (CHAR. VALUE - 33 IN N1; CHAR. IN N2;
            REST OF NAME R.J. IN N2/3);
      EXIT 2;
4;      ERASE; (CLEAR .);
      ZERO; REV; SHLD-8; (SPILL OUT LAST CHAR.);
      SHL+16; (MOVE BACK L.S. END OF NAME);
      REV; SHL-40;
5;      SET B177; AND; (AND OUT M.S.BIT);
      V27P899; =RM7; (SET B/A OF COMMAND
            DICTIONARY);
      SET 65; -; (REDUCE CHAR.RANGE FROM 65-90 TO 0-25);
      SHC-2; DUP;
      =M5; SHL-43; DUP; SHL-1; +; =C7;
      M7M5; (ROUTE WORD);
      SHLC7; SHL-36; (12 BIT ADDRESS IN N1 POINTS
            TO FIRST OCCURENCE OF CHAR. ENTRY IN DICTIONARY);
      DUP; J10=Z; (EXIT 1 IF ZERO ENTRY; NO COMMAND ENDS THUS);
      J6TR;
      PERM; REV; SHLD-16; (R.J.NAME LESS LAST CHAR. IN N1/2);
      CAB;
      J7;
6;      ZERO; NOT; =TR; (RESET TR);
7;      =RM5; M7M5Q; (FETCH M.S.END OF NAME);
      J9*NE;
      REV; M7M5Q;
      J8*NE; ERASE; ERASE; (NAME CLEARED FROM N5);
      M7M5N; (PARAMETER WORD);
      EXIT 3; (SUCCESSFUL SEARCH);
8;      REV; M7M5; (LINK WORD);
11;       DUP; J10=Z;
      J7;
9;      M7M5N; (LINK WORD);
      J11;
10;       EXIT 1; (FAILURE TO FIND ENTRY);
```

```
P824; (USING PARAMETER WORD IN N1 ENTER COMMAND
          CODING);
      (N1=0 FLAG BITS/ENTRY GROUP/DESCRIPTOR MESSAGE ADD.);
      SHL+16; SHL-32;
      DUP; SET 3; SIGN;
      J1*GTZ; (CHECK STACK NOT EXCEEDED);
      SHC-1; DUP; =LINK;
      J2*LTZ;
      EXIT AR3;
2;      EXIT 1 AR3;
£3;     JP840; JP841;
      JP842; JP843;
1;      ERASE;
      V18P899; V17P899; V16P899;
      JS11P841;
          (MESSAGE DECODE FAILS OR OUTPUT);
      EXIT 1;


P828; (INTERROGATIVE MODE MESSAGE);
      DUP; SHL+1; J1*GEZ; (SAVE DATA BUFFER);
      V23P899; =N2; (CURRENT DESK ADDRESS);
      M2TOQ11;
      SET AVOP898; =+111;
      MON11; =Q10; (PACK CONTROL);
      JS1P802; (SAVE INCOMPLETE SECTOR FROM OVERWRITING);
1;      SHL+32; SHL-32;
      JSP826; =Q5;
          (SYSTEM MESSAGE PRESENT; OCW1 = 0; OCW2 =SYSTEM
          MESSAGE ADDRESS IN C-PART);
      EXIT 1;


P827; (SPECIAL TERMINATOR);
          (DUMMY ROUTINE);
      ERASE; ERASE; ERASE; ERASE; =Q5; EXIT 1;
```

```
P826; (CHAIN SYSTEM MESSAGE TO TERMINAL BY PLACING
        MESSAGE DISK ADDRESS IN OCW2 C-PART);
      Q8; Q5; (SAVE);
      CAB;
      SET AVOP809; (BA OF OCW STACK);
      =N5;
      V23P899; SHL+1;
      =M8; (ADDR OF CALLING DESK);
      N5M8N; (OCW2);
      =Q3; =C3;
         (DESCRIPTOR MESSAGE ADDRESS IN C-PART
         OF OCW2);
      ZERO; NOT; =I3; (NO MORE OUTPUT FOR THIS SEQUENCE);
      Q3;
      =M5N8N; (RESTORE UPDATED OCW2);
      ZERO; =M5M8; (OCW1 CLEARED- MESSAGE AVAILABLE);
      =Q5; =Q8; (RESTORE);
      EXIT 1;


P829V4; (CATASTROPHIC SYSTEM FAILURE);
      V0=Q 0/AV1/AV4;
      V1/4=P *Q8SN*ULSYSTEMLFAILURELTYPE;
      V4; OR; =V4;
      V0; SET 8; OUT;
      ZERO; OUT;
```

```
P840; (NEW COMMAND - CLEAR MARKERS FOR TERMINAL);
      SET 6; =C7; (FOR JP811);
      JSP805; (TIME);
      V40P899; V39P899; (DATE);
      V47P899; V46P899; (GOLDSYS 01);
      V23P899;
      SET AVOP801; =M12; (BA OF DESK ASL);
      DUP; =M13; (DESK ADDRESS);
      JS6P801; (CLAIM NEW BLOCK);
      SHL+32;
      =M13M12; (SELECT NEW DISK DATA BLOCK);
      SHL+1; =M7; (*2 FOR P811);
      V48P899; (OCW1);
      JS1P811; (SET UP MESSAGE);
      V34P899; JS3; =V34P899; (RECEIVE STATUS);
      JSP912; JSP913; ERASE; ERASE; ERASE;
            (CLEAR FIFO);
      V23P899;
      JSP806; ERASE;
      ZERO; =MON8; (CLEAR DFW1);
      V23P899; DUP; JSP801;
      SHL-16; DUP; JS1P831;
            (HEADER POINTER FOR INPUT FILE IN
            I-PART DFW1);
      REV;
      DUP; JSP801; SHL-32; CAB; OR;
      =MON8N; (DFW2=0/HEADER(PRESENT)ADDRESS/NOC);
      DUP; =M10;
      SET AVOP898;
      =*M10;
      VOP899; =MOM10; (RESET PACK WORD);
      DUP;
      ZERO; NOT; JS14P841; (CLEAR GOAW);
      =C7;
      V49P899; SHCC7;
      ZERO; NOT; CONT;
      NC7; SHCC7; =V49P899; (SIGN IN);
      EXIT 1;
3;      V23P899; =C7;
4;      SHCC7; STR; ERASE;
      NC7; SHCC7;
      NC7; EXIT 1;
```

```
P841V15; (CALL COMMAND; SET UP JHS; USES DATA BUFFER
          FOR JHS ASSEMBLY AND SHORT PROMPT OUTPUT);
      V0=B 5433 1240 6064 4101;
      V1=B 6442 7240 6224 2305;
      V2=B 2355 2311 6153 5640;
           (12 CHAR. IDENTIF; );
      V3=0 4/6/0; (OCW1);
      V4=B 1273 0662 5014 1510;
      V5=B 2035 1056 5014 5317;
      V6=B 2052 0116 6373 5640;
           (+12 CHAR. JOB NO; );
      V7=B 6514 4515 6132 0314;
      V8=B 6224 6711 6512 0123;
      V9=B 6134 1523 1353 5640;
           (TIME LIMIT SECS.; );
      V10=B 140; (CALL SECTOR CHAIN - CURRENT BLOCK);
      V11=B 2475 2317 6454 2640;
      V12=B 6314 4515 6235 2240;
      V13=B 6575 1104 2473 5640;
           (STORE LIMIT WRDS; );
      V14=B 2415 1311 2355 2240;
      V15=B 5005 4657 2353 5640;
           (PRINT ..... Y/N; );
      JSP911; ZERO; =NON7N; (CLEAR OCW2);
      DUP; (P911 LEAVES CURRENT DESK ADDRESS IN N1);
      JSP806; (DFW1);
      SHC+5; SHL+3; SHL-3; SHC-2; (SET COMMAND NO. 0 IN
           D5-7 DFW1);
      ZERO; NOT; CONT; SHC-3; =NON8;
           (SET PROMPT RESPONSE BIT IN DFW1 D3);
      DUP; =C7; (TERMINAL ADDRESS);
      JSP906; (SET UP Q9 FOR DESK OUTPUT SECTOR B/A);
      JSP910; (CHECK LOCKOUT ON SECTOR);
      ZERO; =E22N9; (CLEAR RESET INHIBIT);
      ZERO; =E11N9; (CLEAR PROMPT SUSPENSION MARKER);
      J21;
1;      (RUNNING ENTRY);
      V23P899; JSP906; (SECTOR B/A);
21;   (INITIAL AND RESET ENTRY);
      E11N9; DUP; NOT; NEG; =E11N9; (UPDATE SUSPENSION MARKER);
      =LINK; EXIT AR 3;
£3;   J4; (INITIAL ENTRY);
      £J5; (PROCESS IDENTIFIER AND ASK FOR JOB NO);
      £J6; (PROCESS JOB NO AND ASK FOR TIME LIMIT);
      £J8; (PROCESS TIME LIMIT AND ASK FOR STORE LIMIT);
      £J15; (PROCESS STORE LIMIT; SET UP JO SECTOR);
      £J22; (FILE OUTPUT OR DIRECT);
4;      V2; V1; V0; (ASK FOR IDENTIFIER);
11;      V23P899; SHL+1; =M7;
      SET 4; =C7;
      V3; (OCW1);
      JSIP811; (SET UP OUTPUT);
```

```
        EXIT 1;
5;        JSP912; JSP913; (GET FIFO);
        =E12M9; =E13M9; =E14M9;
              (IDENTIFIER);
        V6; V5; V4;
        J11; (ASK FOR JOB NO.);
6;        JSP912; JSP913;
        =E15M9; =E16M9; =E17M9;
              (JOB NO.);
19;        V9; V8; V7;
        J11; (ASK FOR TIME LIMIT);
8;        JSP912; JSP913;
        JS6P814; J16; SHL+16;
        =E8M9; (TIME LIMIT);
20;        V13; V12; V11;
        J11; (ASK FOR STORE LIMIT);
16;        JS18; J19;
17;        JS18; J20;
18;        ERASE;
        E11M9; NEG; NOT; =E11M9;
        EXIT 1;
15;        JSP912; JSP913;
        JS8P814; J17; SHL+32;
        =E20M9; (STORE LIMIT);
        ZERO; NOT; DUP;
        SHL=16; (NON SYSTEM MARKER);
13;        SHL+32; E8M9; OR; =E8M9;
              (SET SYS/NON-SYS MARKER);
        =E39M9;
        ZERO; NOT; =E22M9; (INHIBIT RESET);
        ZERO; NOT; SHL+32; =E9M9; (WAITING MARKER);
        JS6P801; DUP; SHL+16;
              (GET OUTPUT FILE);
        V23P899;
        JSP806;
        SHL+8; (CLEAR FLAGS);
        SHL=24; SHC=16;
        OR; =E10M9; (DATA FOC/ OUTPUT FOC/ NO OF DATA SECTORS);
        =E18M9; (FOC OUTPUT FILE IN M-PART);
        M9;
        SET 6; =RC11; M9TOQ11;
        SET 12; =+M9; I9=+1;
£12;        MON9Q; £=MON11Q;  J12C11NZS;
              (MOVE PROG. IDENT + JOB NO. TO
              EQ-S OF DATA BUFFER);
        DUP; =M9;
        DUP; SHL+16; REV;
        SET 39; +; OR;
        V10; JS3P342;
              (JOBORG JHS CHAIN ADDRESS);
        =E38M9; =V10;
        DUP; =E19M9; (SAVE JO SECTOR ADDRESS);
```

```
        REV; =E21M9; (SAVE SECTOR BUFFER PARAMETERS);
26;        V15; REV;
        JSP813; (CONVERT JHS ADDRESS TO TELETYPE
              CHARACTERS);
        V14; J11;
              (PRINT DDDDDD Y/N; QUERY);
22;        JSP912; JSP913;
        ERASE; ERASE;
        SHL+32; SHL-40;
        SET B116; J24=; (N);
        SET B131; J23=; (Y);
        JS18; (RETYPE PROMPT);
        E19M9; J26;
24;        ERASE;
        JSP911; ZERO; =NOM7; (CLEAR OCW1);
        JSP812; ERASE; (ALTER I/O STATUS);
        ZERO; NOT; DUP;
        SHL-16; =NOM7N;
              (SET OCW2=0/-1/-1);
        J25; (SET GOAW ALL ONES);
23;        ERASE; E18M9; (OUTPUT FOC);
25;        E21M9; (SECTOR CORE BUFFER);
        E19M9; (JO SECTOR ADDRESS);
        SHL+32; DUP; PERM; OR;
        SET 47; OUT; ZERO; =TR;
        SET 41; OUT; (WRITE JHS TO NOC ON DISK QUEUE);
        SET -4; OUT; (INITIALISE OR UPDATE DIRECTOR
              JOB QUEUE FOR THIS STACK);
        JSP911; ZERO; =NOM7; (CLEAR OCW1);
        DUP; JSP806;
        ERASE;
        ZERO; =NOM8;
        REV; DUP; =NOM7N; (OCW2=FOC OUTPUT OR -1
              IF NO ONLINE OUTPUT REQUESTED);
14;        REV; SET AVOP896;
        =M15; =M14;
              (M14 = TERMINAL ADDRESS);
        =M14M15; (SET GOAW= DATA FOC);
        V49P899; JSOP840; =V49P899; (FORCE LOG OUT);
        EXIT 1;
```

```
P842V5; (PROMPT COMMAND SET UP JO SECTOR);
        V0=B 2164 0531 1403 0055;
        V1=B 1322 6455 2525 0264;
        V2=0;
              (GAY0Q----UP4);
        V3=B 1265 4460 5423 0060;
        V4=B 1473 2055 2105 1525;
        V5=0;
              (+Y01Q034-DSU);
        V23P899; JSP806;
        SHC+5; SHL+3; SET 2; OR; SHC-5;
              (PUT COMMAND NO. 2 IN D5-7 DFW1);
        ZERO; NOT; CONT; SHC-3; =NOM8;
        V23P899; DUP; =C7; JSP906;
        JSP910; (TEST SUSPENSION ON LOCKOUT - DOES
              NOT RETURN IF LOCKED OUT);
        SET 5; =E11N9; (SYSTEM PROMPT SUSPENSION MARKER
              IN P841 - CALL);
        V0; =E12N9; V1; =E13N9;
        V2; =E14N9; V3; =E15N9;
        V4; =E16N9; V5; =E17N9;
        SET 9999; SHL+16; =E8N9; (TIME LIMIT);
        SET 12800; SHL+32; =E20N9; (STORE LIMIT);
        ZERO; NOT; DUP;
        J13P841; (SET MARKERS IN P841);
3;        SHC-4; ZERO; SHLD+4; DUP;
        SET 15; SIGN; J1*GEZ; (CHECK SECTOR ADDRESS
              NOT *GE 15);
        SHLD-4;
        ERASE; SHC+4;
        DUP; NOT; NEG;
              (NOC/NOC/PRESENT OC);
2;        DUP; EXIT 1;
1;        SHLD-4; ERASE; SHC+4;
              (GET NEW BLOCK);
        JS6P801; J2;
```

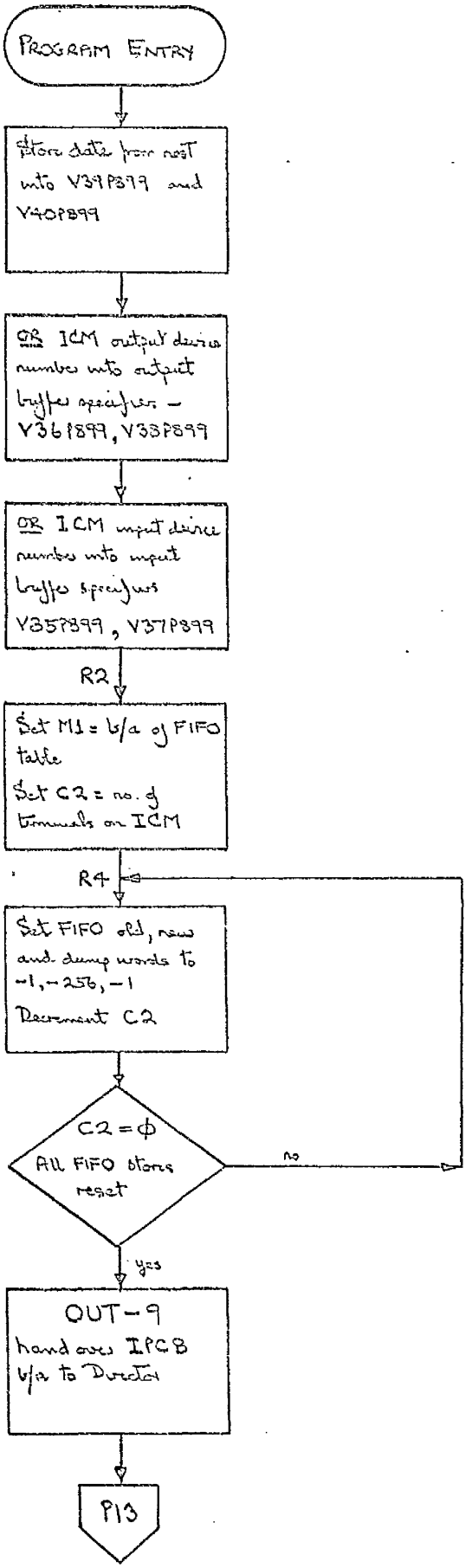
```
P843V2; (LIST COMMAND);
        V0=B 6314 4523 6512 0306;
        V1=B 6234 6305 5004 7125;
        V2=B 2324 1305 6453 5640;
              (LIST FILE NUMBER; );
        V23P899; DUP; JSP906; (DATA BUFFER
              B/A IN M9);
        DUP; =C8;
        V3P899; SHLC8; J8*LTZ; (J IF COMMAND ALREADY SUSPENDED);
        JSP806;
        SHC+5;
        SHL+2; ZERO; NOT; CONT; SHL=2;
        SHC=2; ZERO; NOT; CONT; SHC=3;
              (SET COMMAND NO. 1 IN D5-7; RESPONSE FLAG IN DFW1);
        =NOM8; (COMMAND SELECT AND PARAMETER FLAG);
3;        V2; V1; V0;
        J11P841; (ASK FOR FILE NO);
8;        ERASE;
1;        (RUNNING ENTRY);
        V23P899; JSP906; (SET UP M9);
        JSP912; E0M7; E1M7; E2M7; (GET FIFO);
        JS12P814; J4; (CONVERT TO BINARY);
        DUP; =I15; (SAVE SECTOR ADDRESS);
        SHL=4;
        SET 46; *DIV1; =C15;
        =M15;
        SET AV8P801; =M14;
        M14M15; SHCC15;
        J3*GEZ; (CHECK BLOCK ASSIGNED);
        M9; DUP; SHL+16;
        REV; SET 39; +;
        OR;
        I15; SHL+32; OR;
32;       V23P899; =C7; ZERO; =TR;
        DUP; =Q1;
        TLOQ1; J30TR; (J IF LOCKOUT ON);
        C7; JSP806; DUP; J31*LTZ; (J IF TRANSFER STARTED);
        ZERO; NOT; CONT; =NOM8; (DO DFW1 = 1 = TRANSFER STARTED);
        JS33; (CLEAR FIFO);
        SET 47; OUT; ZERO; =TR;
        SET 42; OUT; (READ DOWN JHS SECTOR);
        ZERO; (DUMMY);
30;       JS2P804; (SET COMMAND SUSPENSION MARKER);
        ERASE; EXIT 2; (LEAVE D2/3 DFW1;V3P899 SET);
31;       STR; ERASE; =NOM8; (CLEAR TRANSFER MARKER);
        ERASE;
        JS1P804; (CLEAR COMMAND SUSPENSION MARKER);
        E9N9; SHA=32; NOT;
              (IF 0 THEN JOB DONE);
        J7*NEZ;
        V31P899; V30P899; V29P899;
        V23P899; JSP806; ERASE;
```

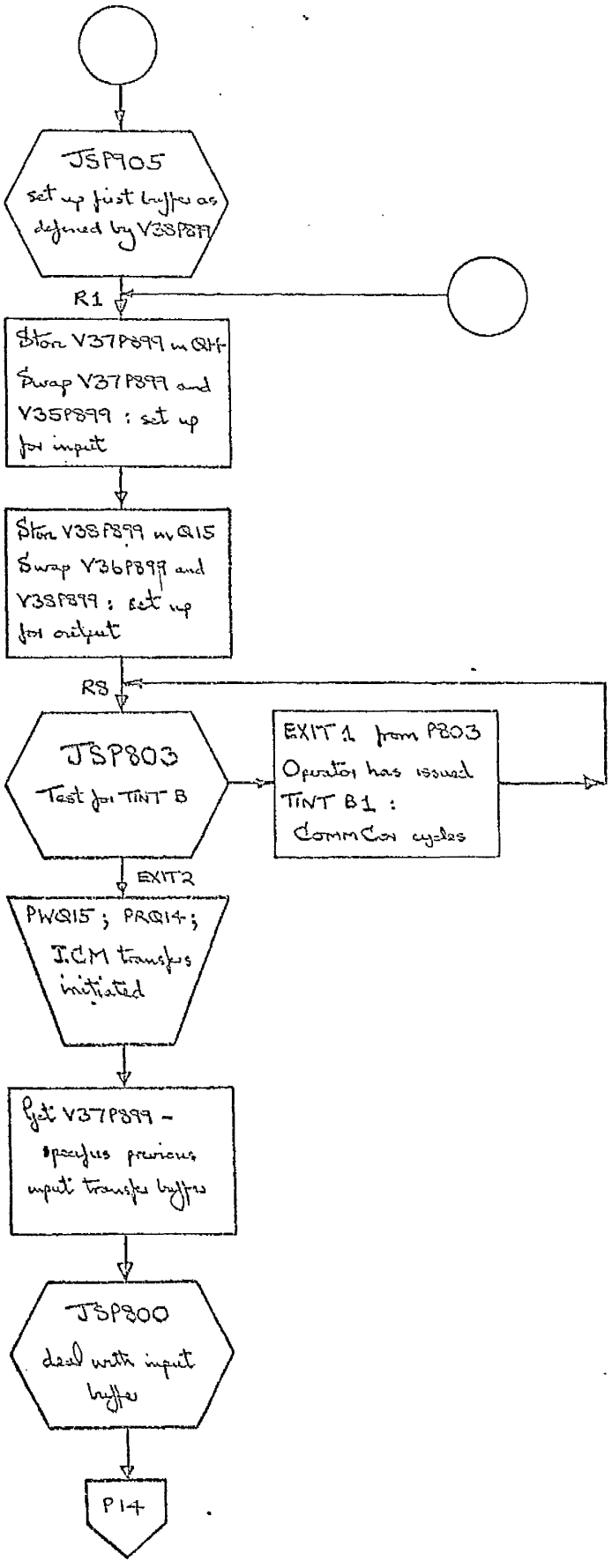
```
        ZERO; =MON8; (CLEAR DF%1);
        J11P841; (OUTPUT JOB STILL QUEUED MESSAGE);
7;        JSP911; ZERO; =MON7; (CLEAR OCW1);
        E10M9; SHL+16; SHL-32; (FOC OUTPUT);
        DUP; SET 3072; SIGN; J5*GTZ;
        SET 16; SHL+16; OR; DUP; =MON7N;
              (OCW2 = 0/16/FOC OUTPUT);
        REV; DUP; JSP806; ERASE;
        ZERO; =MON8; (CLEAR DF%1);
        REV;
        J14P841; (SET GOAW);
5;        ERASE;
4;        ERASE; JS33; J3;
33;        JSP912; JSP913;
        ERASE; ERASE; ERASE;
        EXIT 1; (CLEAR FIFO);
```

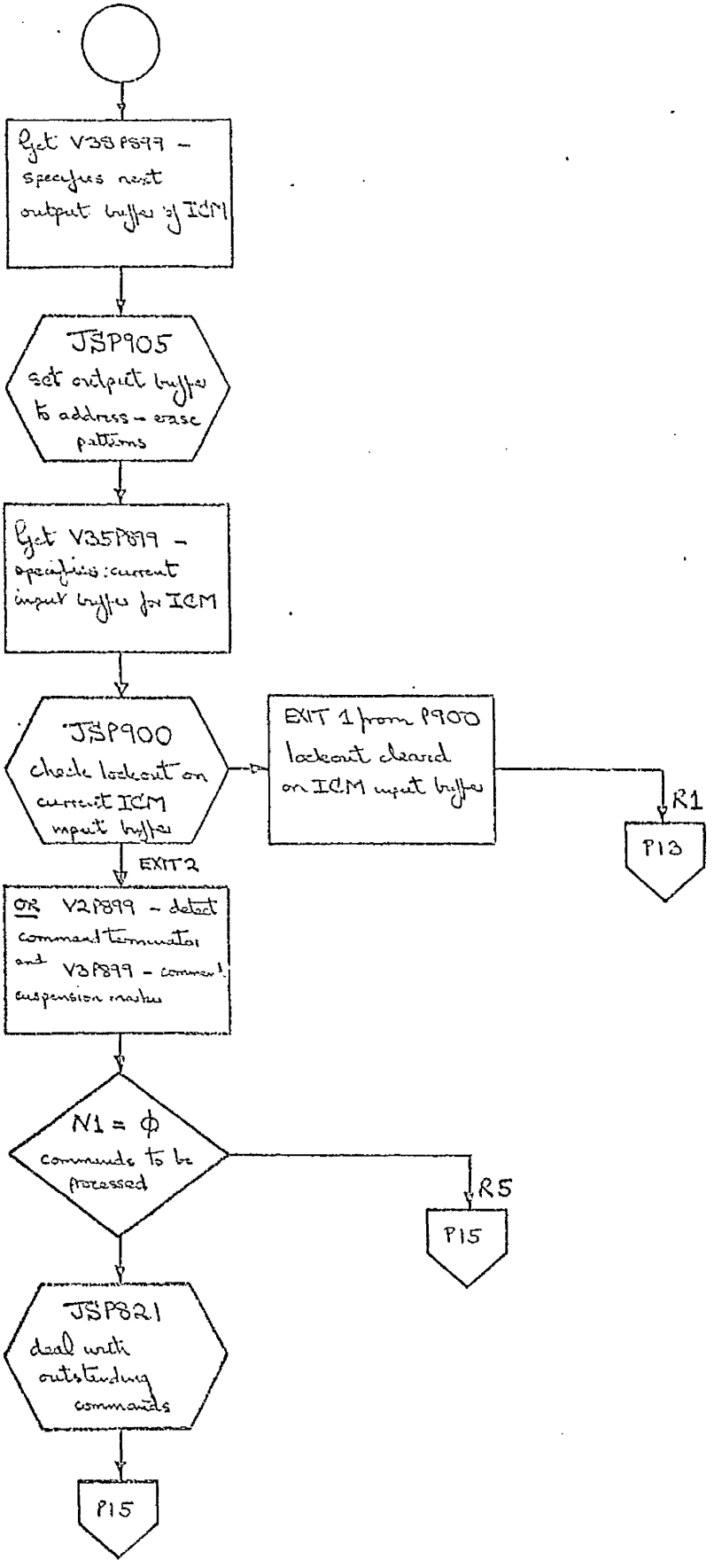
```
P899V50; (CONSTANTS AND COMMUNICATIONS AREA);
      V0=Q 30/1/0;   (CONTROL PACKING INTO SECTOR BUFFER);
      V1=Q 3/6/0;      (FIFO DELETE OCW1);
      V2=0; (PO MARKER);
      V3=0; (PO MARKER);
      V4=B 6224 7126 2034 6311;
      V5=B 2112 0303 6364 6515;
      V6=B 2024 7104 4320 5012;
           (INVALID COMMAND);
      V7=B 6145 2516 2345 4640;
      V8=B 6314 4516 2272 0327;
      V9=B 6375 1104 4320 5012;
           (FUNNY LINK WORD);
      V14=Q 0/0/AY168;       (B/A OF DESK FLAG WORD TABLE);
      V15=Q 0/0/AY320;   (BA OF OUTPUT SECTOR STACK);
      V16=B 2114 2703 6364 2305;
      V17=B 5014 3101 6234 6123;
      V18=B 5003 0261 4320 5012;
           (DECODE FAILS 01);
      V20=B 4320 5012 5252 0240; (TELETYPE CR LF LF  SP SP);
      V21=Q 0/0/AY184;       (B/A OF FIFO TABLE);
      V23=0;              (P821 ADDRESS OF TERMINAL);
      V24= -256;             (DUMMY DUMP WORD FOR P822);
      V25=0;              (M.S. COMMAND NAME);
      V26=0;              (L.S. COMMAND NAME);
      V27=Q 0/0/AYOP897;       (COMMAND DICTIONARY B/A);
      V29=B 6254 7502 5005 1724;
      V30=B 6234 6314 5015 0525;
      V31=B 6125 2705 2110 6412; (JOB STILL QUEUED);
      V34=-1;              (TERMINAL I/O STATUS; 1 RECEIVE);
      V35=Q 0/AY224/AY233;    (INPUT);
      V36=Q 0/AY256/AY265;    (OUTPUT);
      V37=Q 0/AY288/AY297;    (OUTPUT);
      V38=Q 0/AY0/AY9;    (OUTPUT);
      V39=0;              (DATE 1);
      V40=0;              (DATE 2);
      V41=B 0377 0777 1377 1777;   (ADDRESS ERASE GROUPS);
      V43=4;              (NO. OF TERMINALS);
      V46=B 2174 7714 2105 1531;       (GOLDSY);
      V47=B 2472 0060 5512 0240;   (S 04   );
      V48=Q 6/6/0;        (OCW1 FOR NEW COMMAND);
      V49=0;              (SIGN-IN MARKER);
```

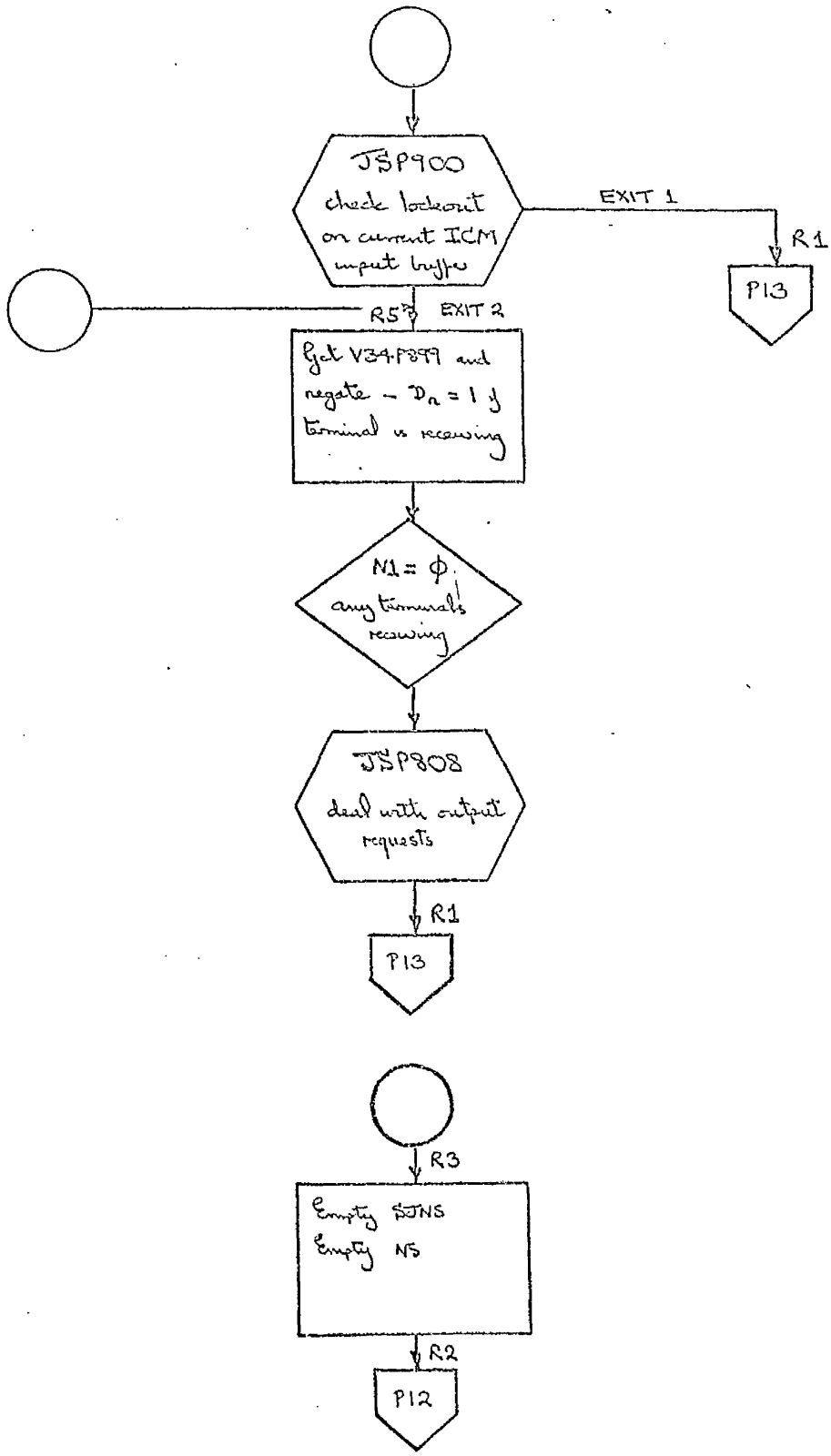
```
P898V15; (SECTOR PACK CONTROL WORDS);


P897V46; (COMMAND DICTIONARY);
      V0=0;                    (A,B,C,D);
      V1=0;                 (E,F,G,H);
      V2=B 0000 0000 0000 0013;       (I,J,K,L);
      V3=0;                    (M,N,O,P);
      V4=B 0000 0000 0000 0017;       (Q,R,S,T);
      V5=B 0000 0000 0007 0000;       (U,V,W,X);
      V6=0;                    (Y,Z,=,-);
      V7=0;                    (EMPTY);
      V8=B 47305;              (NE);
      V9=0;                    (LINK TO NEXT ENTRY);
      V10=Q 0/0/1;    (Q FLAG BITS/ P824 ENTRY/ ADDRESS OF
                        DESCRIPTOR SECTOR);
      V11=0;
      V12=B 6064 0714;       (CAL);
      V13=0;
      V14=Q B 40000/1/2;
      V15=0;                    (EMPTY);
      V16=B 5 0322 6364 6520;       (PROMP);
      V17=19;
      V18=Q B 40000/2/3;
      V19=0;
      V20=B 6314 4523;    (LIS);
      V21=0;
      V22=Q 0/3/4;


P896V31; (GOLD AND DIRECTOR OUTPUT AVAILABILITY
           WORD TABLES GOAW/DOAW);
      V0=-1;
      V1=-1;
      V2=-1;
      V3=-1;
      V4=-1;
      V5=-1;
      V6=-1;
      V7=-1;
      V8=-1;
      V9=-1;
      V10=-1;
      V11=-1;
      V12=-1;
      V13=-1;
      V14=-1;
      V15=-1; (END OF GOAW LIST);
```

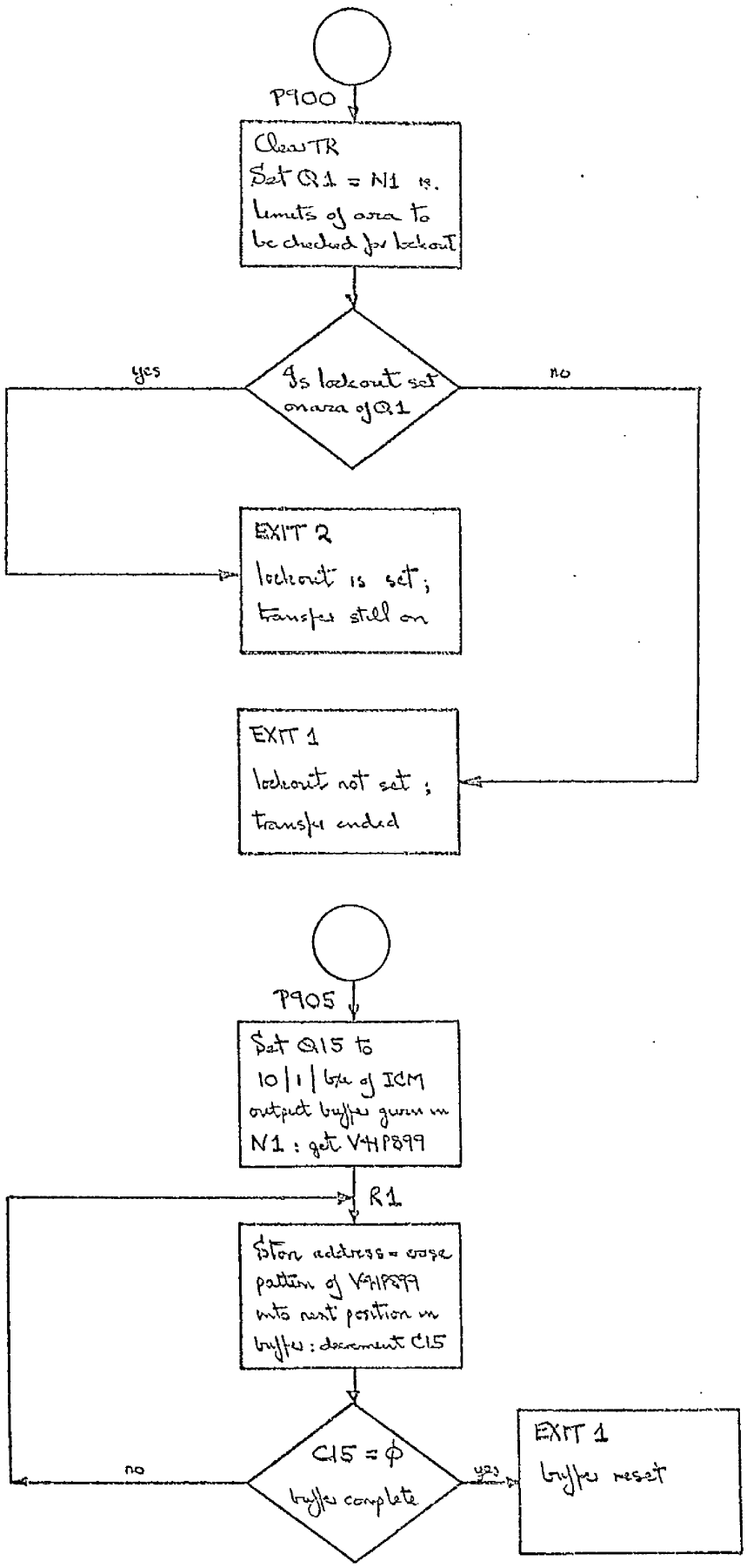
```
P999V0; (L1 DRIVER);


L1V89:      (MONITOR AT SET TEST POINTS,15,12,64);
      ERASE;
      EXIT 1; (DUMMY L1 TEST PACK);


FINISH;
```

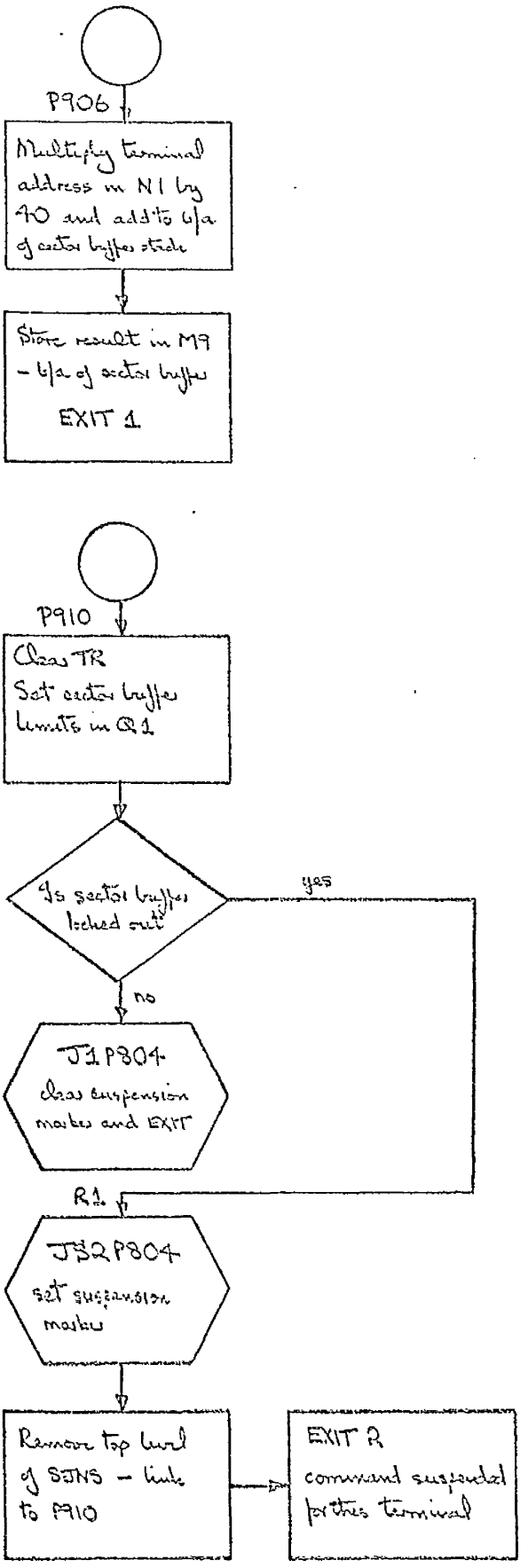

PROGRAM ENTRY
Store data from next into V39P899 and V40P899
OR ICM output device number into output buffer specifier - V36P899, V38P899
OR ICM input device number into input buffer specifier V35P899, V37P899
R2
Set M1 = b/a of FIFO table
Set C2 = no. of terminals on ICM
R4
Set FIFO old, new and dump words to -1, -256, -1
Decrement C2
C2 = ϕ
All FIFO stores reset
no
yes
OUT-9
Hand over IPC8 b/a to Director
P13

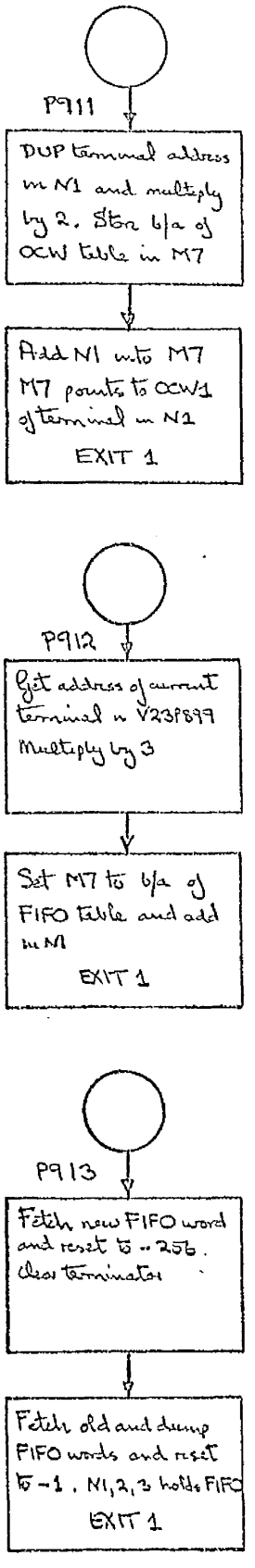

JSP905
set up first buffer as defined by V38P899
R1
Store V37P899 in Q14
Swap V37P899 and V35P899 : set up for input
Store V38P899 in Q15
Swap V36P899 and V38P899 : set up for output
R8
JSP803
Test for TINT B
EXIT 1 from P803
Operator has issued TINT B1 : CommCon cycles
EXIT 2
PWQ15 ; PRQ14 ;
I.C.M transfers initiated
Get V37P899 - specifies previous input transfer buffer
JSP800
deal with input buffer
P14

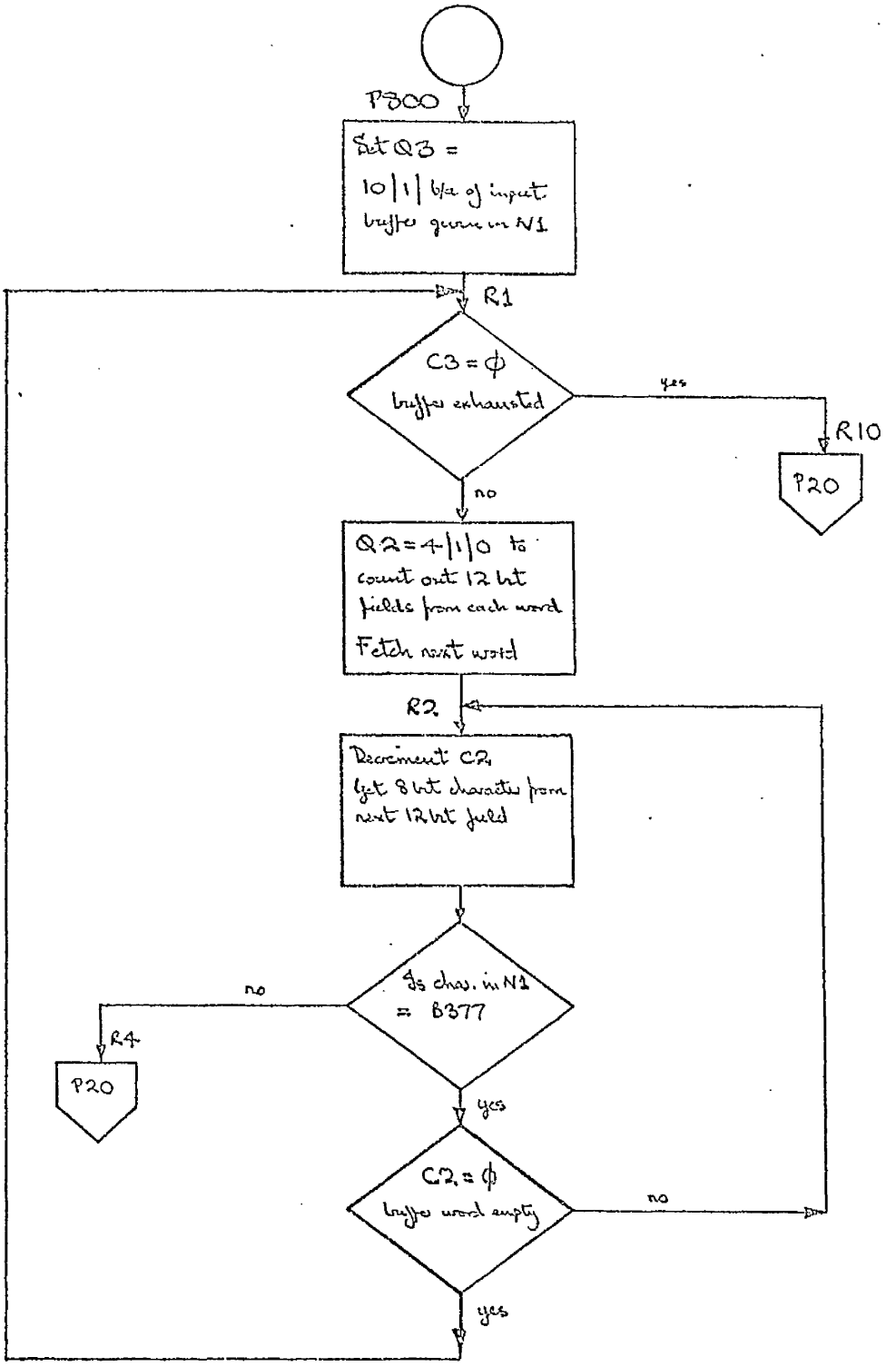

Get V38P899 - specifies next output buffer for ICM
JSP905
set output buffer to address - erase patterns
Get V35P899 - specifies current input buffer for ICM
JSP900
check lockout on current ICM input buffer
EXIT 1 from P900
lockout cleared on ICM input buffer
R1
P13
EXIT 2
OR V21899 - detect command termination and V38P899 - command suspension status
N1 = ϕ
commands to be processed
R5
P15
JSP821
deal with outstanding commands
P15

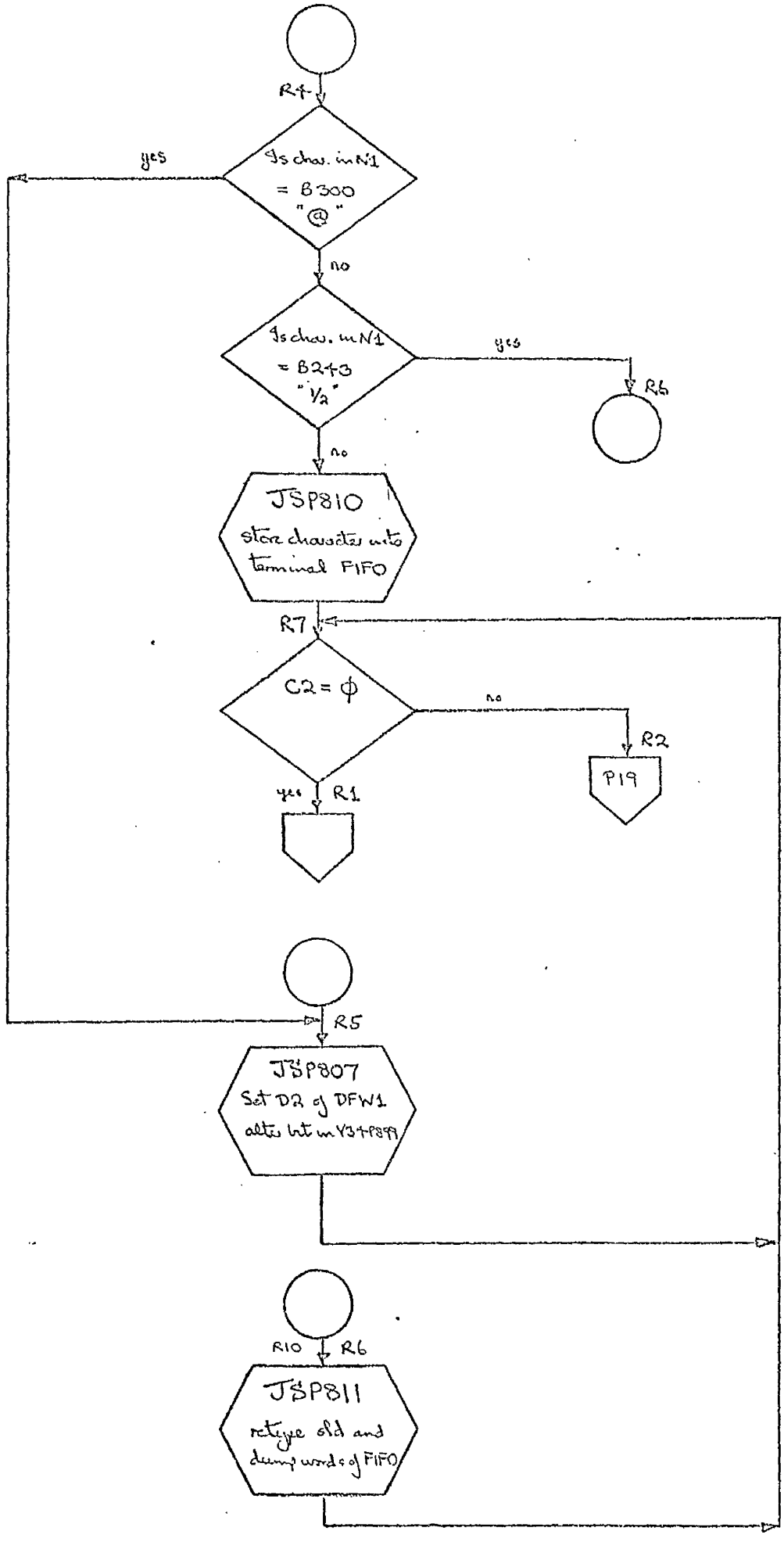

JSP900
check lockout on current ICM input buffer
EXIT 1
R1
P13
R5
EXIT 2
Get V34-P899 and negate - D_n = 1 if terminal is receiving
N1 = ϕ
any terminals receiving
JSP808
deal with output requests
R1
P13
R3
Empty SDNS
Empty NS
R2
P12

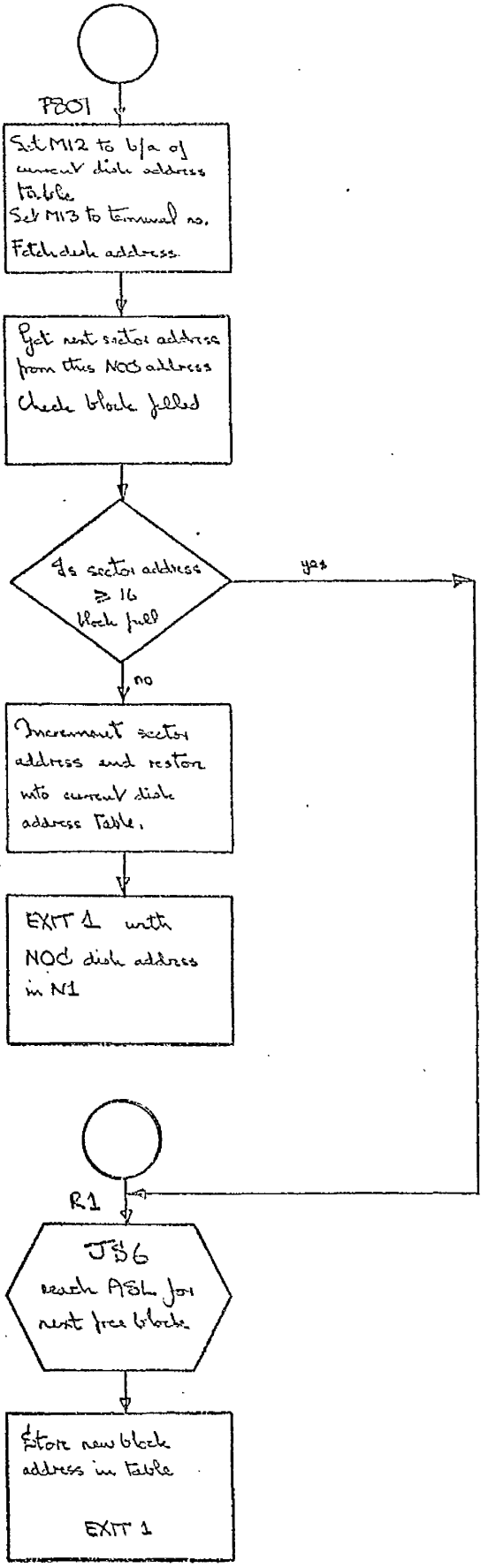

P900
Clear TR
Set Q1 = N1 is limits of area to be checked for lockout
Is lockout set on area of Q1
yes
no
EXIT 2
lockout is set; transfer still on
EXIT 1
lockout not set; transfer ended
P905
Set Q15 to 10/1/ byte of ICM output buffer given in N1 : get V+1P899
R1
Store address = coupe pattern of V+1P899 into next position in buffer : decrement C15
C15 = ϕ buffer complete
no
yes
EXIT 1
buffer reset

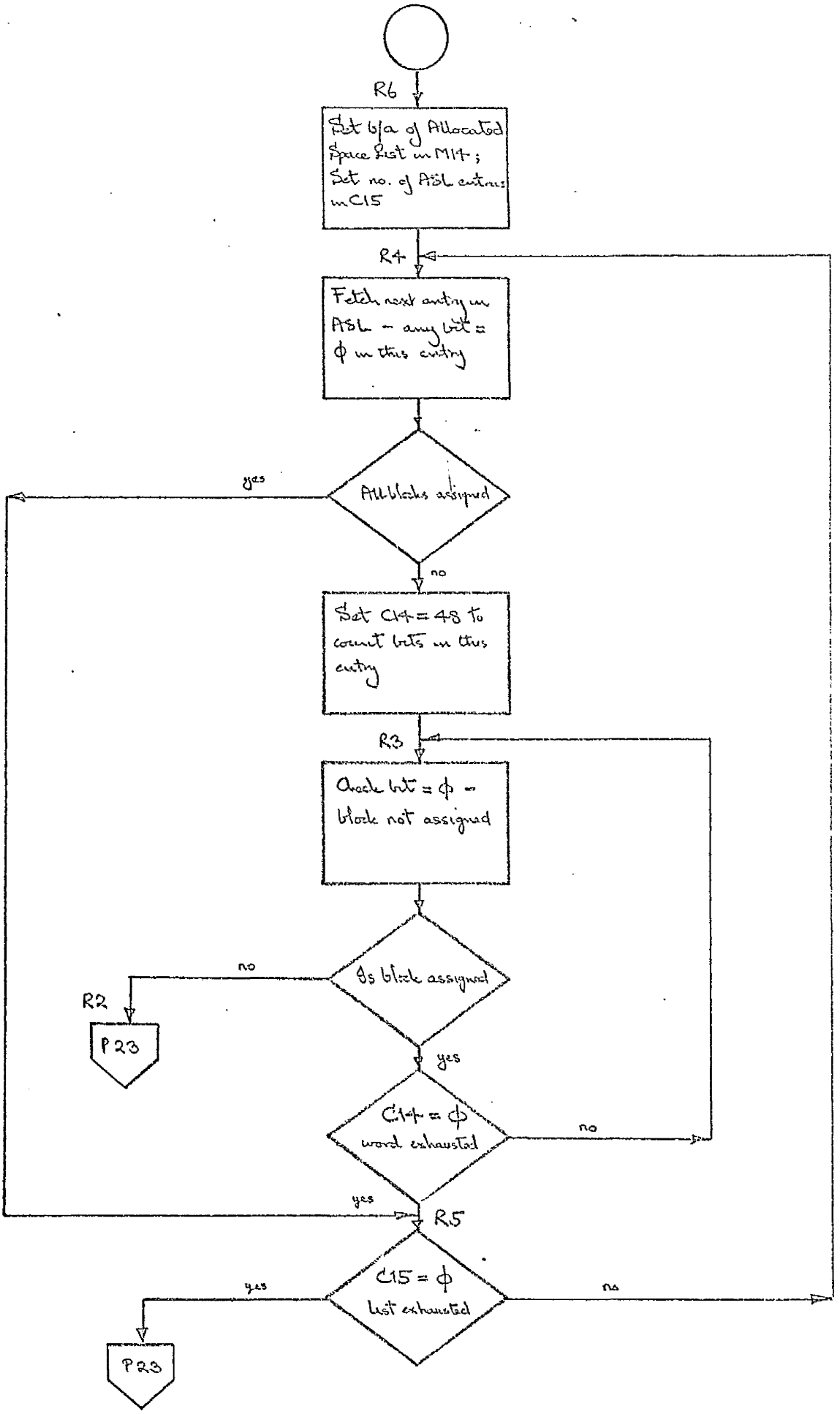

P906
Multiply terminal address in N1 by 40 and add to b/a of sector buffer stack
Store result in M9 - b/a of sector buffer
EXIT 1
P910
Clear TR
Set sector buffer limits in Q1
Is sector buffer locked out?
yes
no
J1 P804
clear suspension marker and EXIT
R1
J52 P804
set suspension marker
Remove top level of SJNS - link to P910
EXIT 2
command suspended for this terminal

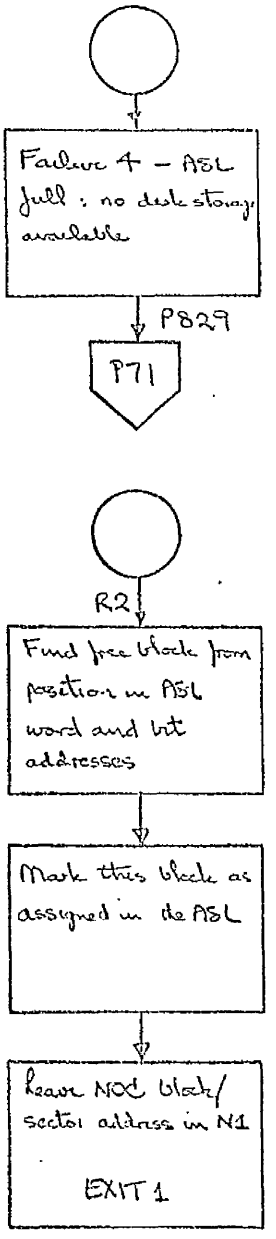

P911
DUP terminal address in N1 and multiply by 2. Stor b/a of OCW table in M7
Add N1 into M7
M7 points to OCW1 of terminal in N1
EXIT 1
P912
Get address of current terminal in V23F899
Multiply by 3
Set M7 to b/a of FIFO table and add in M1
EXIT 1
P913
Fetch new FIFO word and reset to -256. clear terminator
Fetch old and dump FIFO words and reset to -1. N1,2,3 holds FIFO
EXIT 1

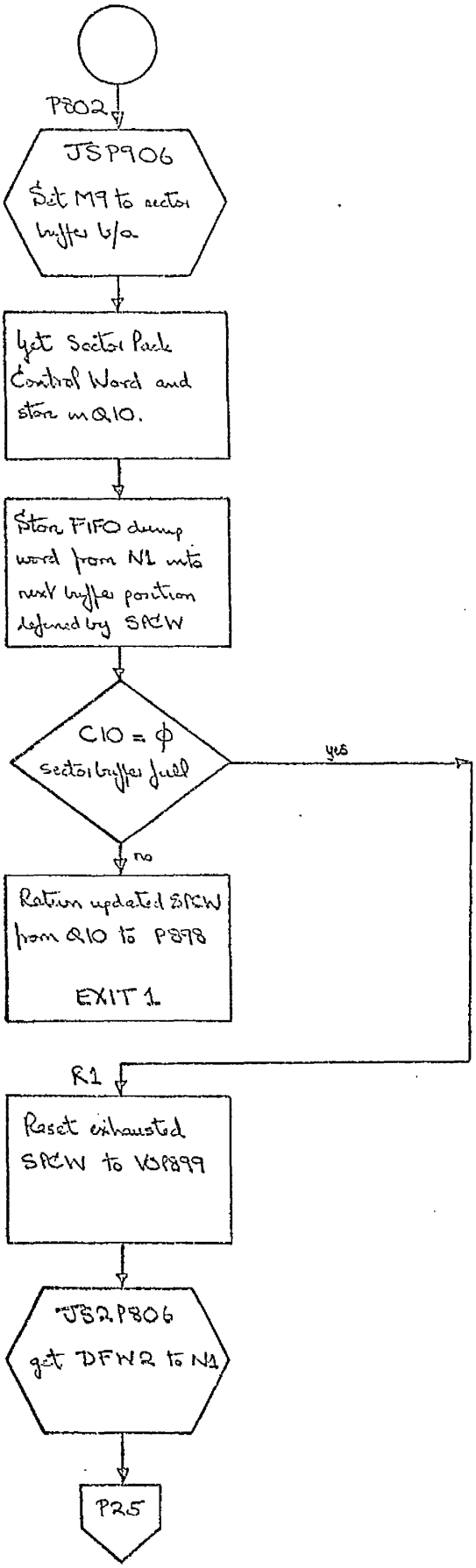

P300
Set Q3 =
10|1| bfr of input
buffer queue in N1
R1
C3 = ϕ
buffer exhausted
yes
R10
P20
no
Q2 = 4|1|0 to
count out 12 bit
fields from each word
Fetch next word
R2
Decrement C2
Get 8 bit character from
next 12 bit field
Is char. in N1
= 8377
no
R4
P20
yes
C2 = ϕ
buffer word empty
no
yes

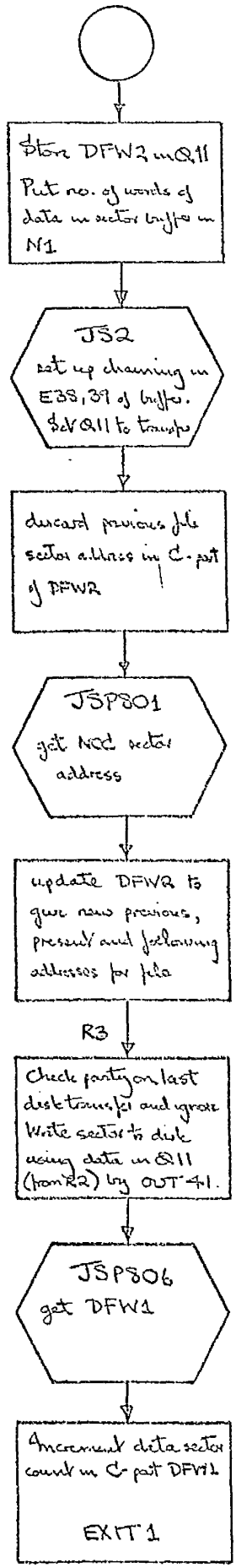

R4
Is char. in N1 = B300 "@"
yes
no
Is char. in N1 = B243 "Ya"
yes
R6
no
JSP810
store character into terminal FIFO
R7
C2 = ϕ
no
R2
P19
yes
R1
R5
JSP807
Set D2 of DFW1 alter bit in Y3+P899
R10
R6
JSP811
retype old and dump words of FIFO

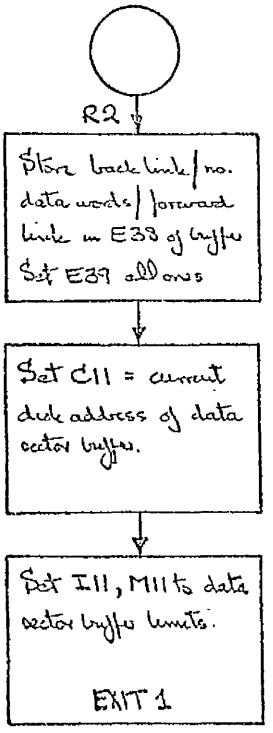

P201
Set M12 to b/a of current disk address table
Set M13 to terminal no.
Fetch disk address
Get next sector address from this NOC address
Check block filled
Is sector address ≥ 16 block full
yes
no
Increment sector address and restore into current disk address table.
EXIT 1 with NOC disk address in N1
R1
JSB
search ASL for next free block
Store new block address in table
EXIT 1

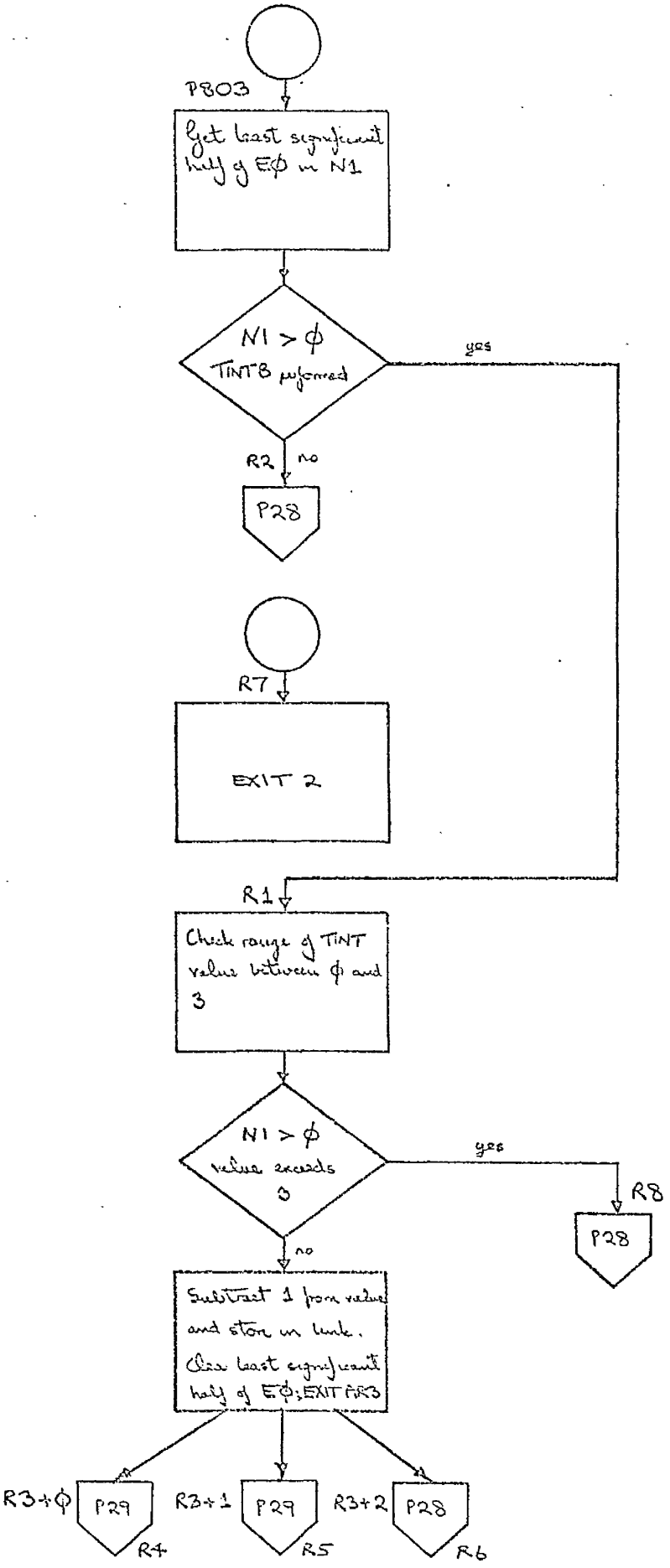

R6
Set b/a of Allocated Space list in M14; Set no. of ASL entries in C15
R4
Fetch next entry in ASL - any bit = ϕ in this entry
All blocks assigned
yes
no
Set C14 = 48 to count bits in this entry
R3
Check bit = ϕ - block not assigned
Is block assigned
no
R2
P23
yes
C14 = ϕ word exhausted
no
yes
R5
C15 = ϕ list exhausted
yes
P23
no

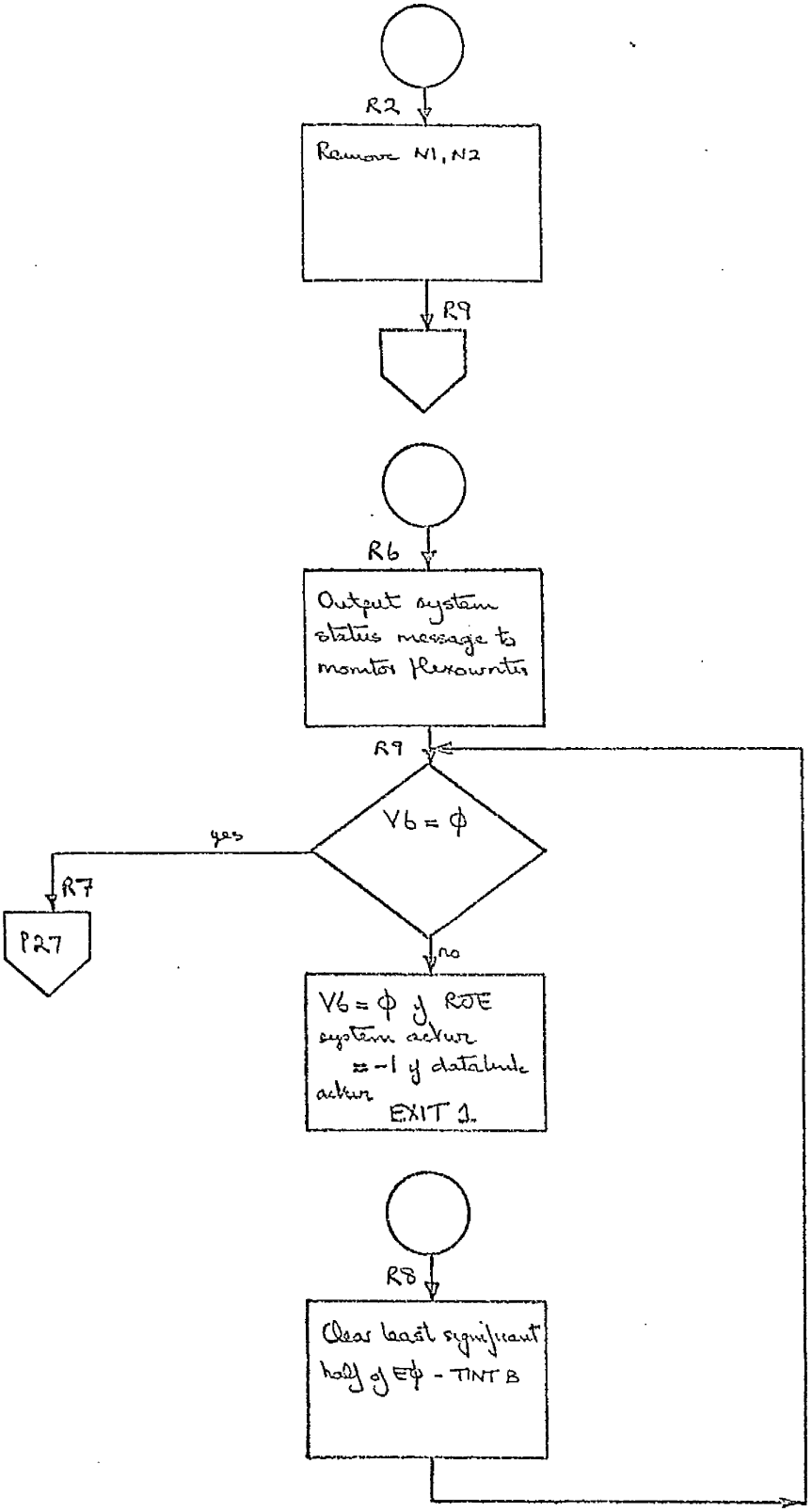

Failure 4 – ASL full : no disk storage available
P829
P71
R2
Find free block from position in ASL word and bit addresses
Mark this block as assigned in the ASL
Leave NOC block/ sector address in N1
EXIT 1

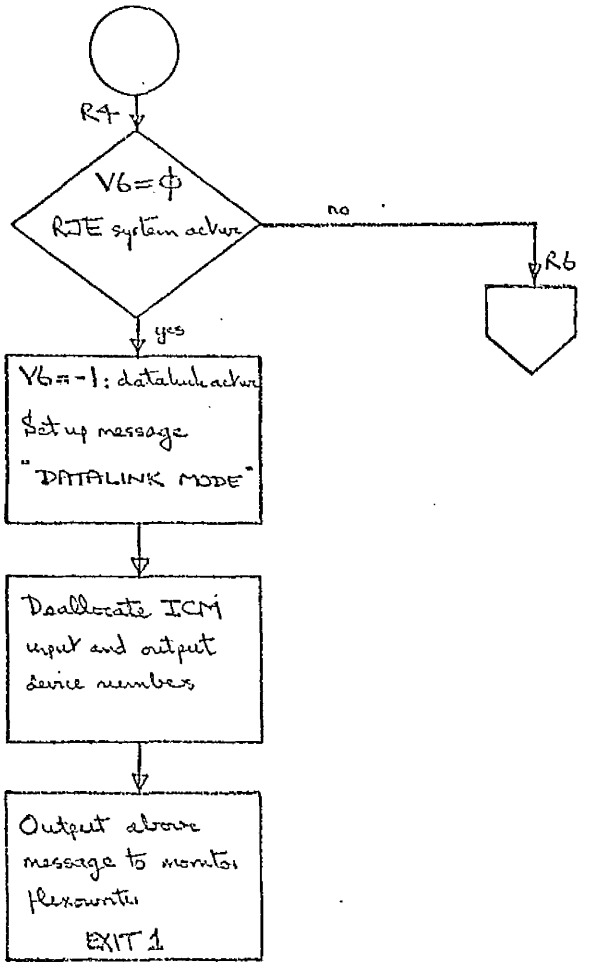

P802
JSP906
Set M9 to sector buffer b/a
Get Sector Pack Control Word and store in Q10.
Store FIFO dump word from N1 into next buffer position defined by SPCW
C10 = ϕ sector buffer full
yes
no
Return updated SPCW from Q10 to P878
EXIT 1.
R1
Reset exhausted SPCW to V01899
JSP806
get DFW2 to N1
P25

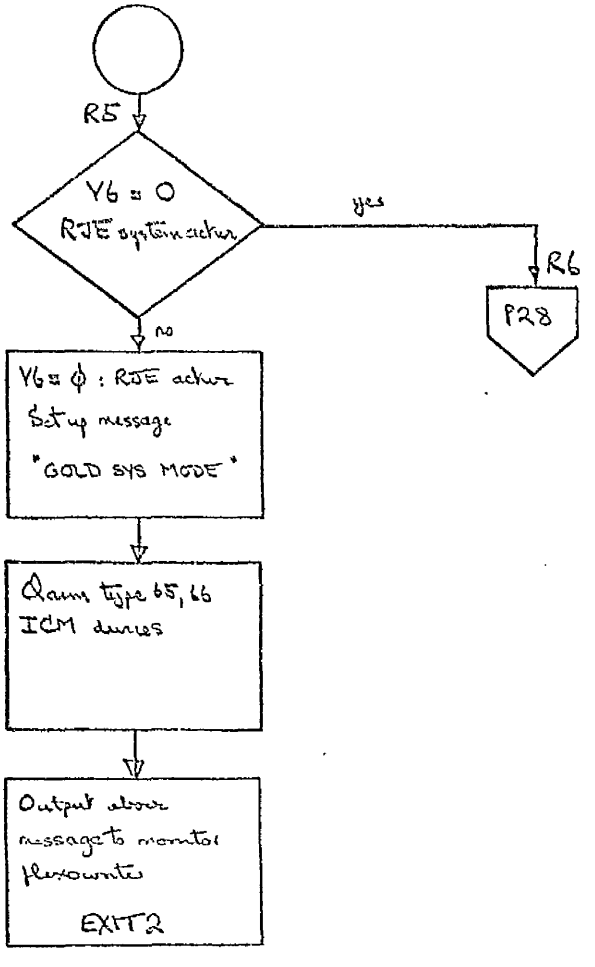

Store DFW2 in Q11
Put no. of words of data in sector buffer in N1
JS2
set up chaining in E38,39 of buffer. Set Q11 to transfer
discard previous file sector address in C-part of DFW2
JSP801
get NCC sector address
update DFW2 to give new previous, present and following addresses for file
R3
Check parity on last disk transfer and ignore
Write sector to disk using data in Q11 (from R2) by OUT 41.
JSP806
get DFW1
Increment data sector count in C-part DFW1
EXIT 1

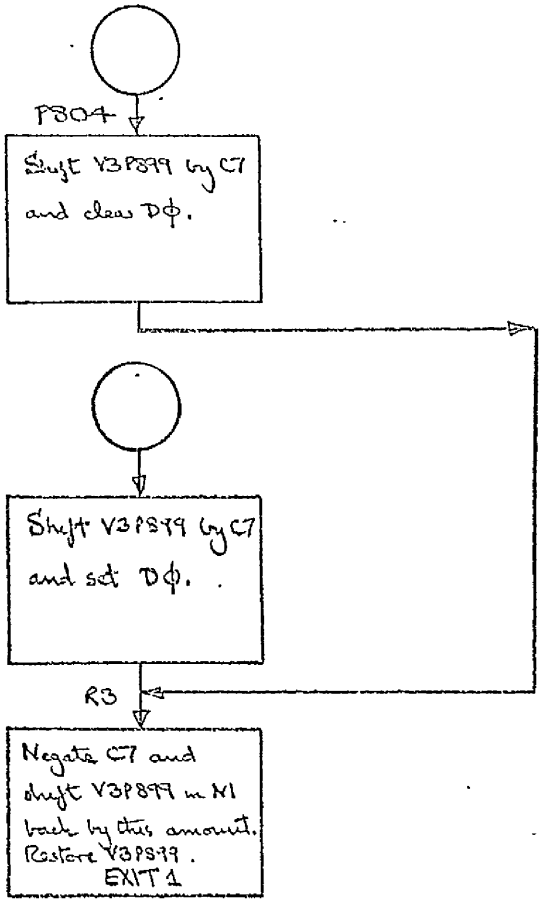

R2
Store back link / no. data words / forward link in E38 of buffer
Set E39 all ones
Set C11 = current disk address of data sector buffer.
Set I11, M11 to data sector buffer limits.
EXIT 1

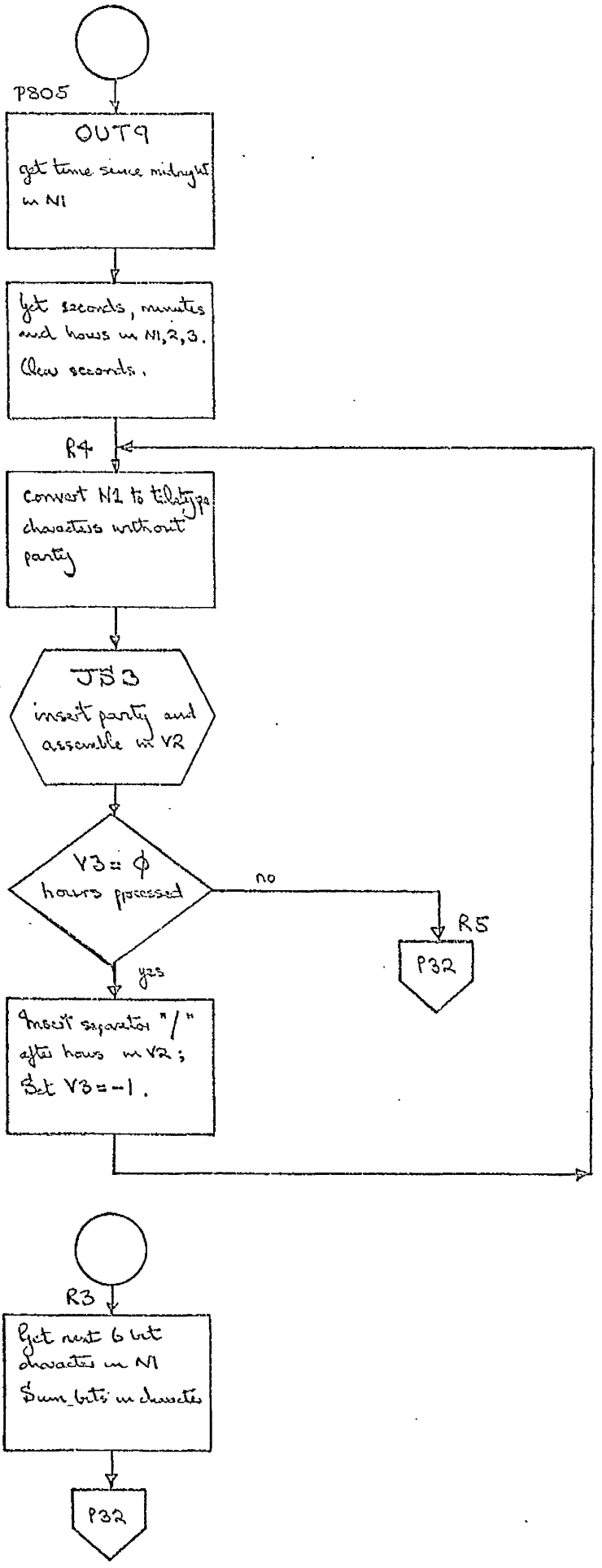

P803
Get least significant half of EΦ in N1
N1 > Φ
TINTB performed
yes
R2
no
P28
R7
EXIT 2
R1
Check range of TINT value between Φ and 3
N1 > Φ
value exceeds 3
yes
R8
P28
no
Subtract 1 from value and store in link. Clear least significant half of EΦ; EXIT AR3
R3+Φ
P29
R4
R3+1
P29
R5
R3+2
P28
R6

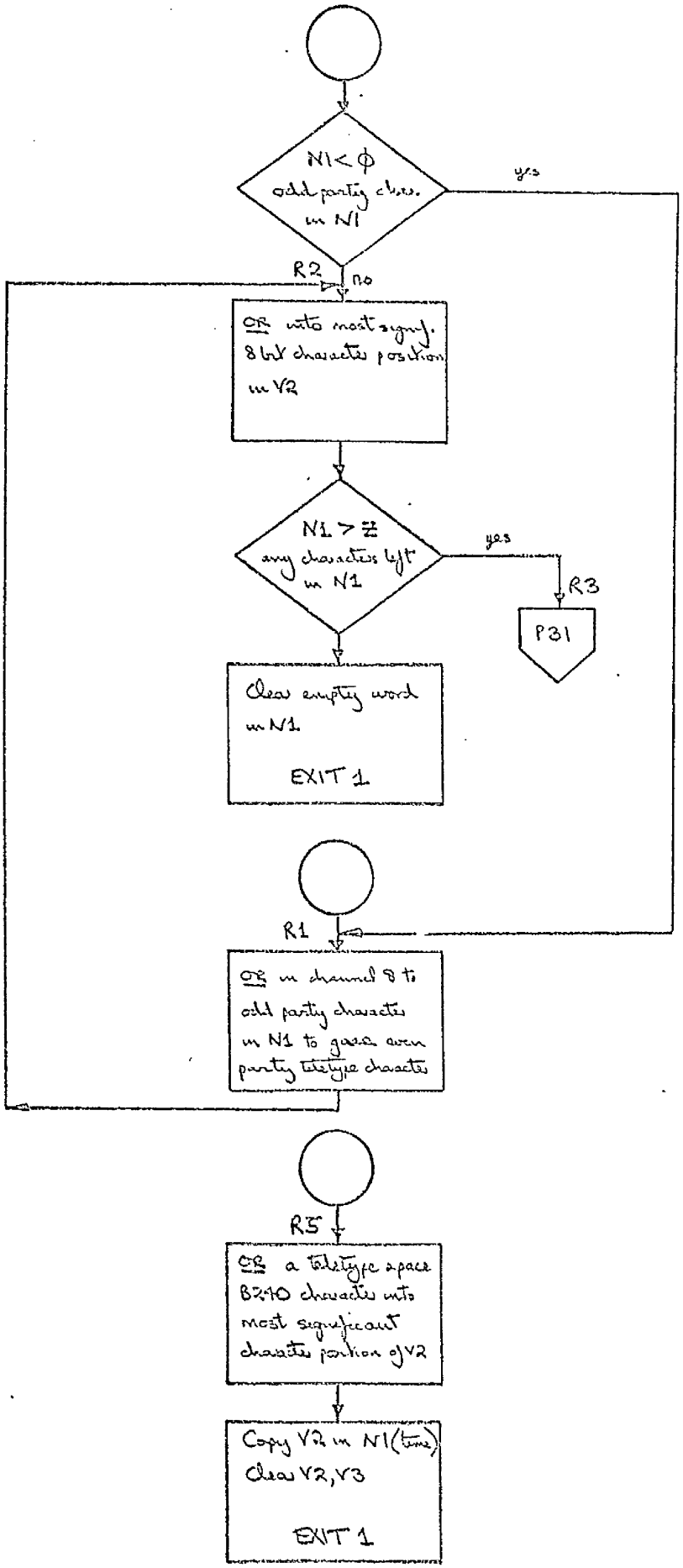

R2
Remove N1, N2
R9
R6
Output system status message to monitor Resources
R9
V6 = ϕ
yes
R7
P27
no
V6 = ϕ & RQE system active = -1 & data link active
EXIT 1.
R8
Clear least significant half of Eϕ - TINT B

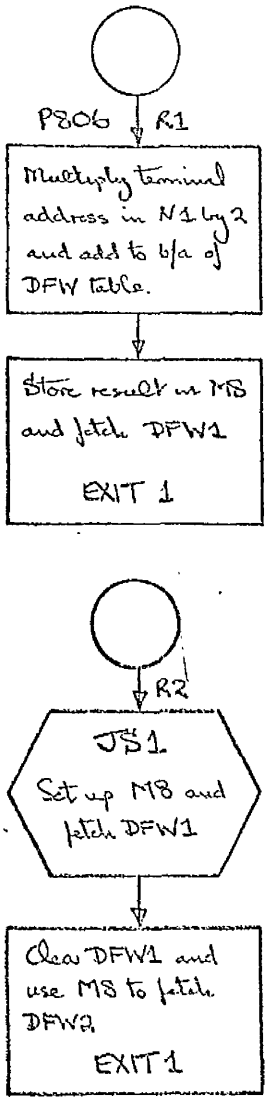

R4
V6=ϕ
RJE system active
no
R6
yes
V6=-1: datalink active
Set up message
"DATALINK MODE"
Deallocate ICM
input and output
device numbers
Output above
message to monitor
flexowriter
EXIT 1

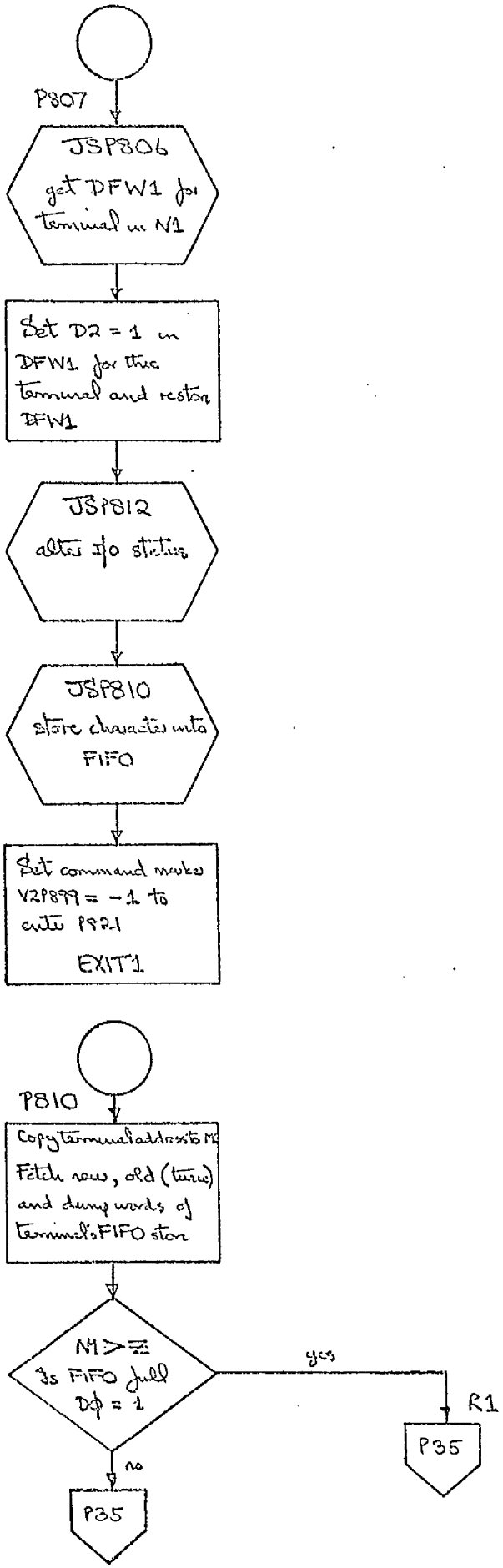

R5
V6 = 0
RJE system active
yes
R6
P28
no
V6 = ϕ : RJE active
Set up message
"GOLD SYS MODE"
Query type 65, 66
ICM devices
Output above
message to monitor
flexowriter
EXIT 2

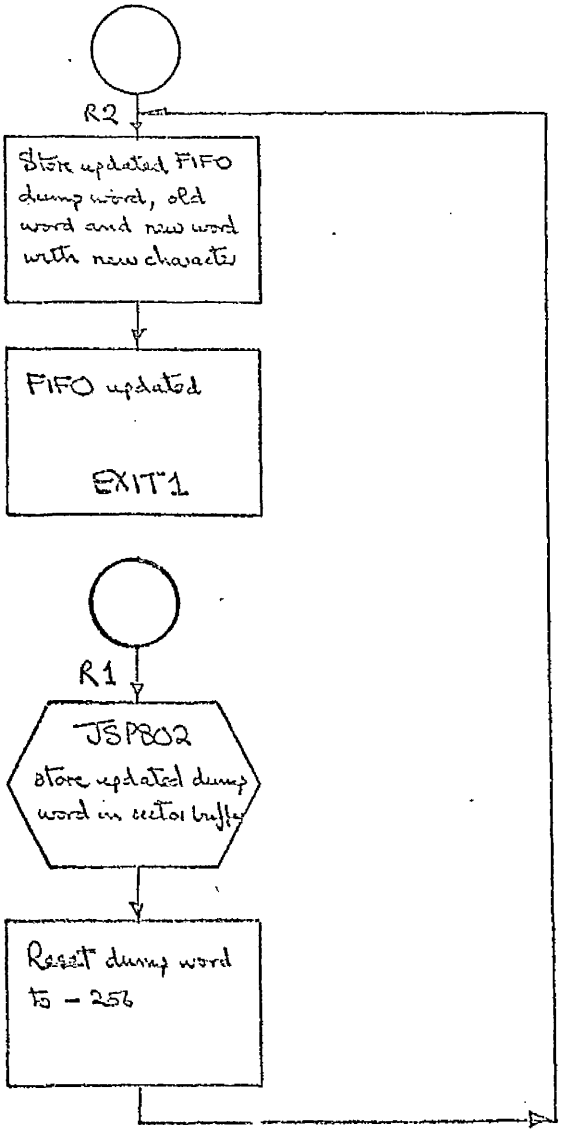

P804
Shift V3P899 by C7 and clear Dϕ.
Shift V3P899 by C7 and set Dϕ.
R3
Negate C7 and shift V3P899 in M1 back by this amount. Restore V3P899.
EXIT 1

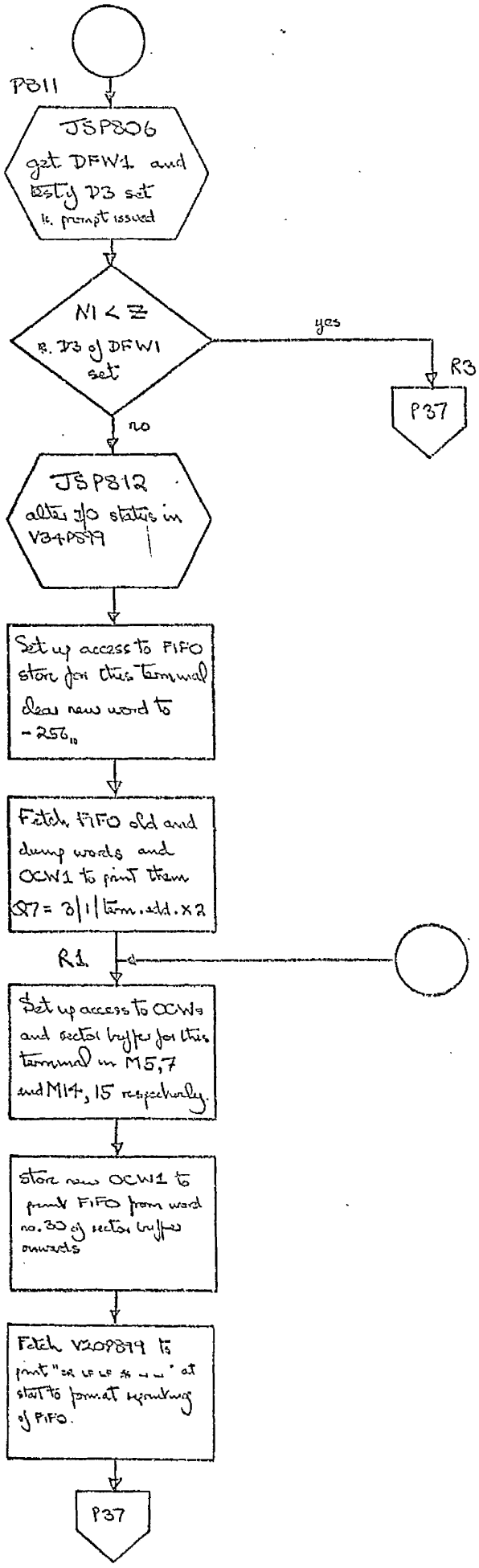

P805
OUT9
get time since midnight in N1
get seconds, minutes and hours in N1,2,3.
Clear seconds.
R4
convert N1 to teletype characters without parity
JS3
insert parity and assemble in V2
V3 = ϕ
hours processed
no
R5
P32
yes
insert separator "/" after hours in V2;
Set V3 = -1.
R3
Get next 6 bit character in N1
Sum_bits in character
P32

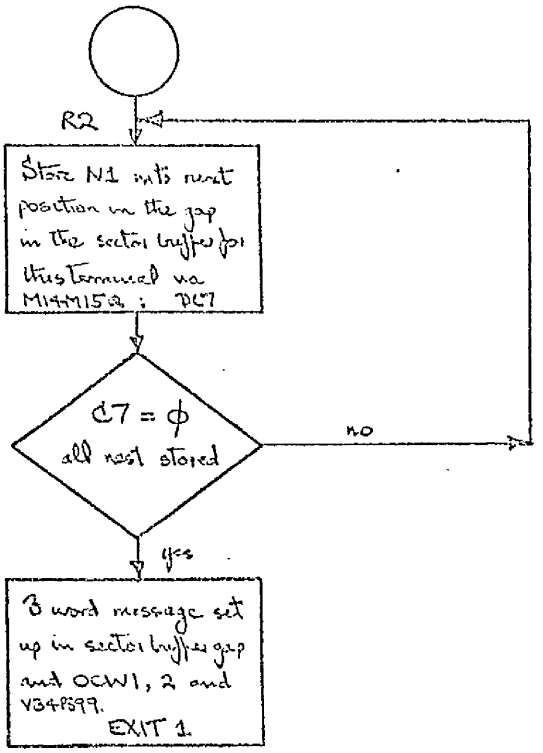

N1 < ϕ
odd parity chars. in N1
yes
R2
no
OR into most signif. 8 bit character position in V2
N1 > ϕ
any characters left in N1
yes
R3
P31
Clear empty word in N1.
EXIT 1
R1
OR in channel 8 to odd parity character in N1 to generate even parity teletype character
R5
OR a teletype space 8210 character into most significant character portion of V2
Copy V2 in N1 (time)
Clear V2, V3
EXIT 1

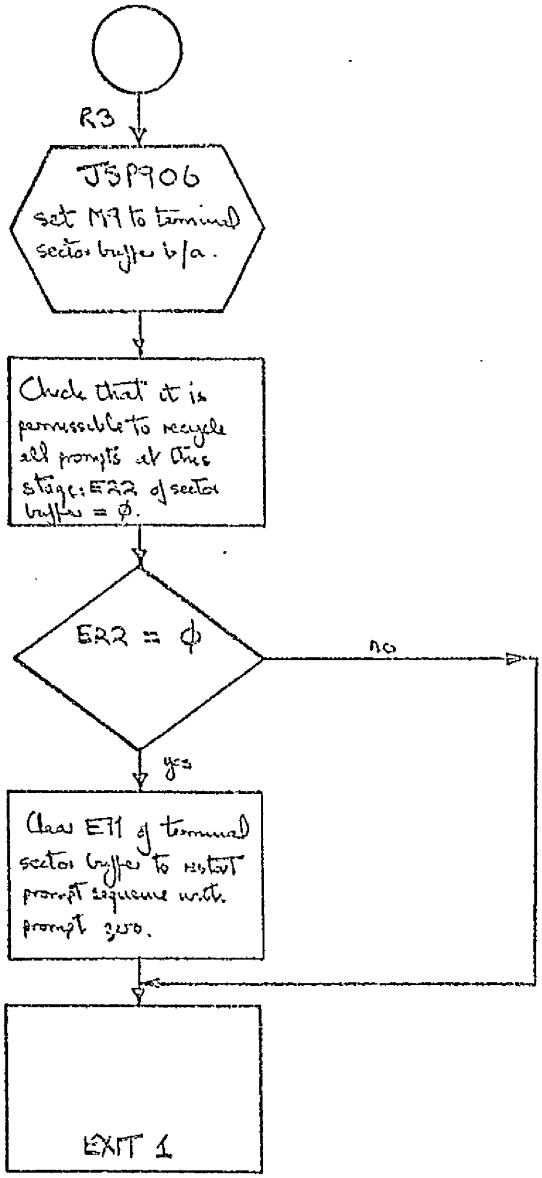

P806
R1
Multiply terminal address in N1 by 2 and add to b/a of DFW table.
Store result in M8 and fetch DFW1
EXIT 1
R2
JS1.
Set up M8 and fetch DFW1
Clear DFW1 and use M8 to fetch DFW2.
EXIT 1

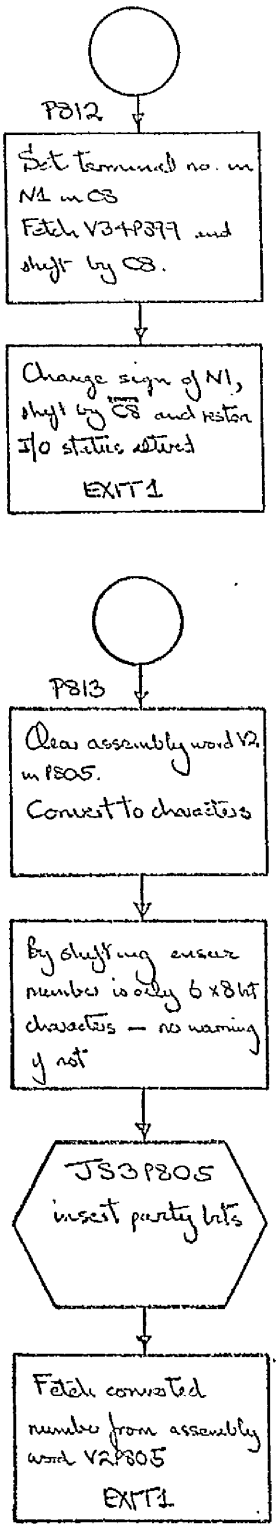

P807
JSP806
get DFW1 for terminal in N1
Set D2 = 1 in DFW1 for the terminal and restore DFW1
JSP812
alter I/O status
JSP810
store characters into FIFO
Set command marker V2P899 = -1 to exit P92.1
EXIT1
P810
Copy terminal address to M
Fetch new, old (turn) and dump words of terminal's FIFO stor
N1 >= ?
Is FIFO full D6 = 1
yes
R1
P35
no
P35

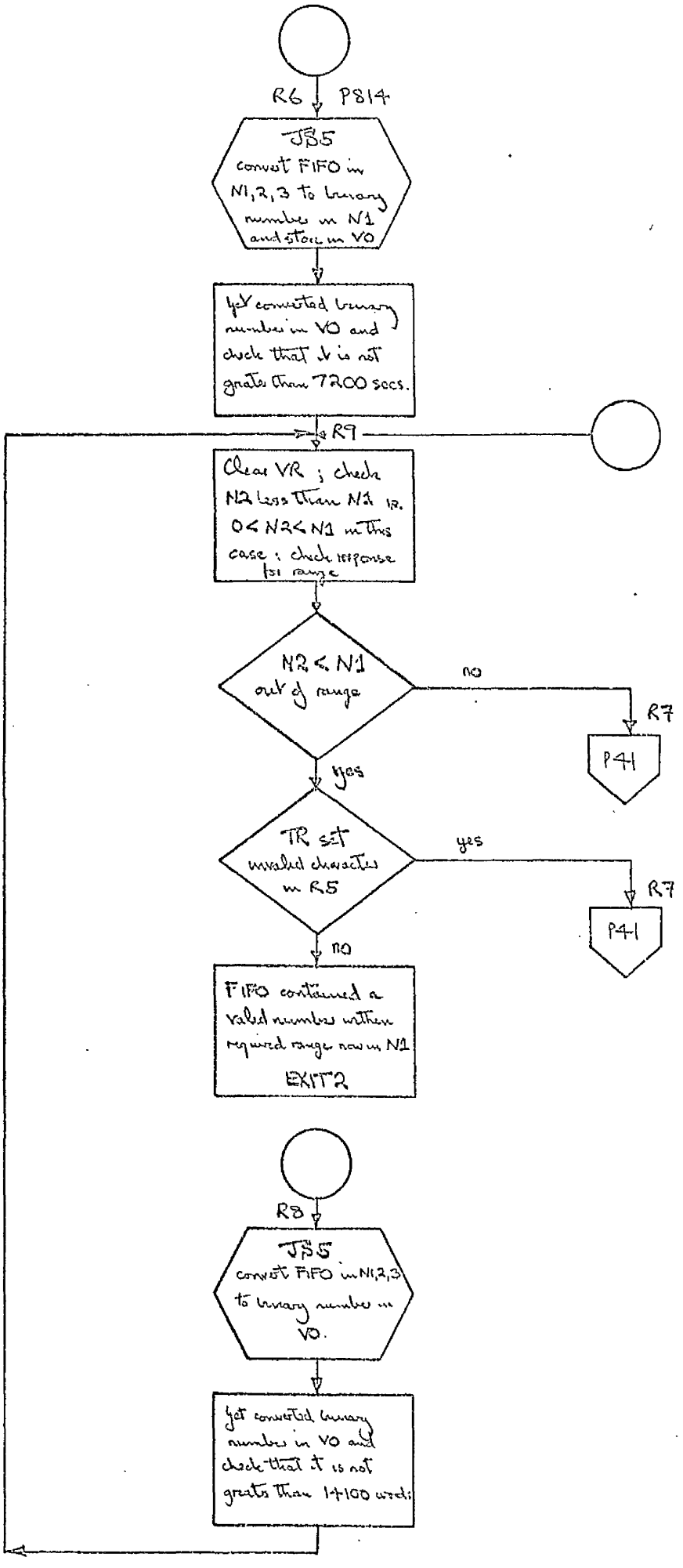

R2
Store updated FIFO dump word, old word and new word with new characte
FIFO updated
EXIT 1
R1
JSP802
store updated dump word in vector buffe
Reset dump word to - 256

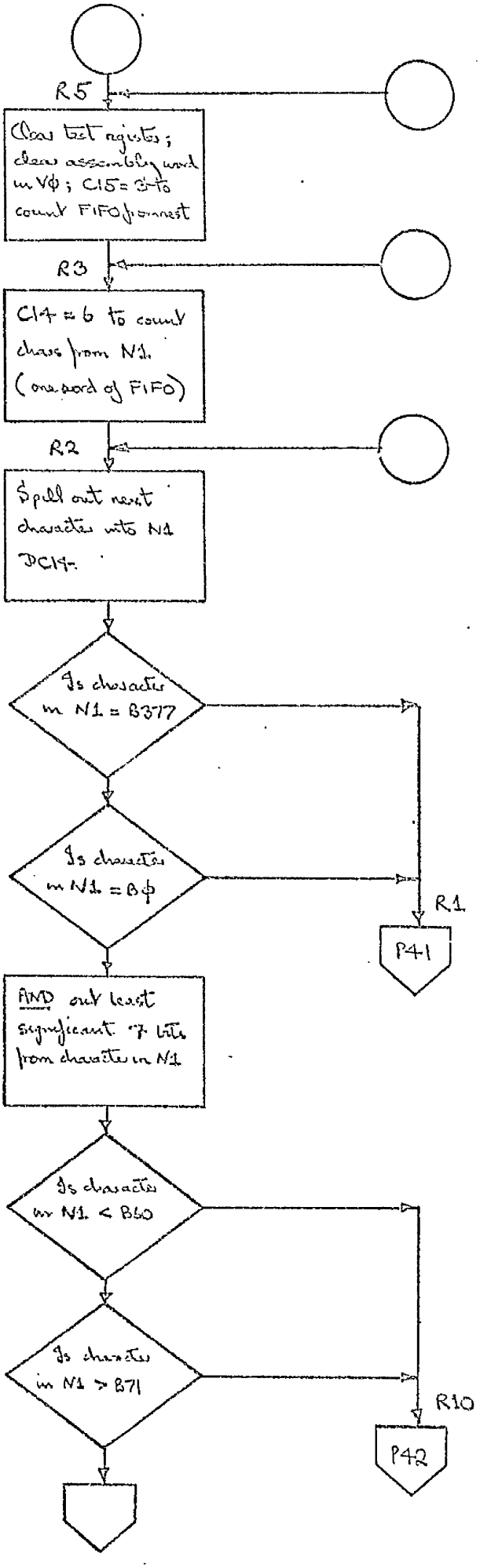

P811
JSP806
get DFW1 and testy D3 set
ie. prompt issued
N1 < 2
ie. D3 of DFW1 set
yes
R3
P37
no
JSP812
alter I/O status in V34P899
Set up access to FIFO store for this terminal
clear new word to -256₁₀
Fetch FIFO old and dump words and OCW1 to print them
S7 = 3/1/term.add.X2
R1
Set up access to OCWs and sector buffer for this terminal in M5,7 and M14, 15 respectively
store new OCW1 to print FIFO from word no. 30 of sector buffer onwards
Fetch V20P899 to print "CR LF LF % - -" at start to format reporting of FIFO.
P37

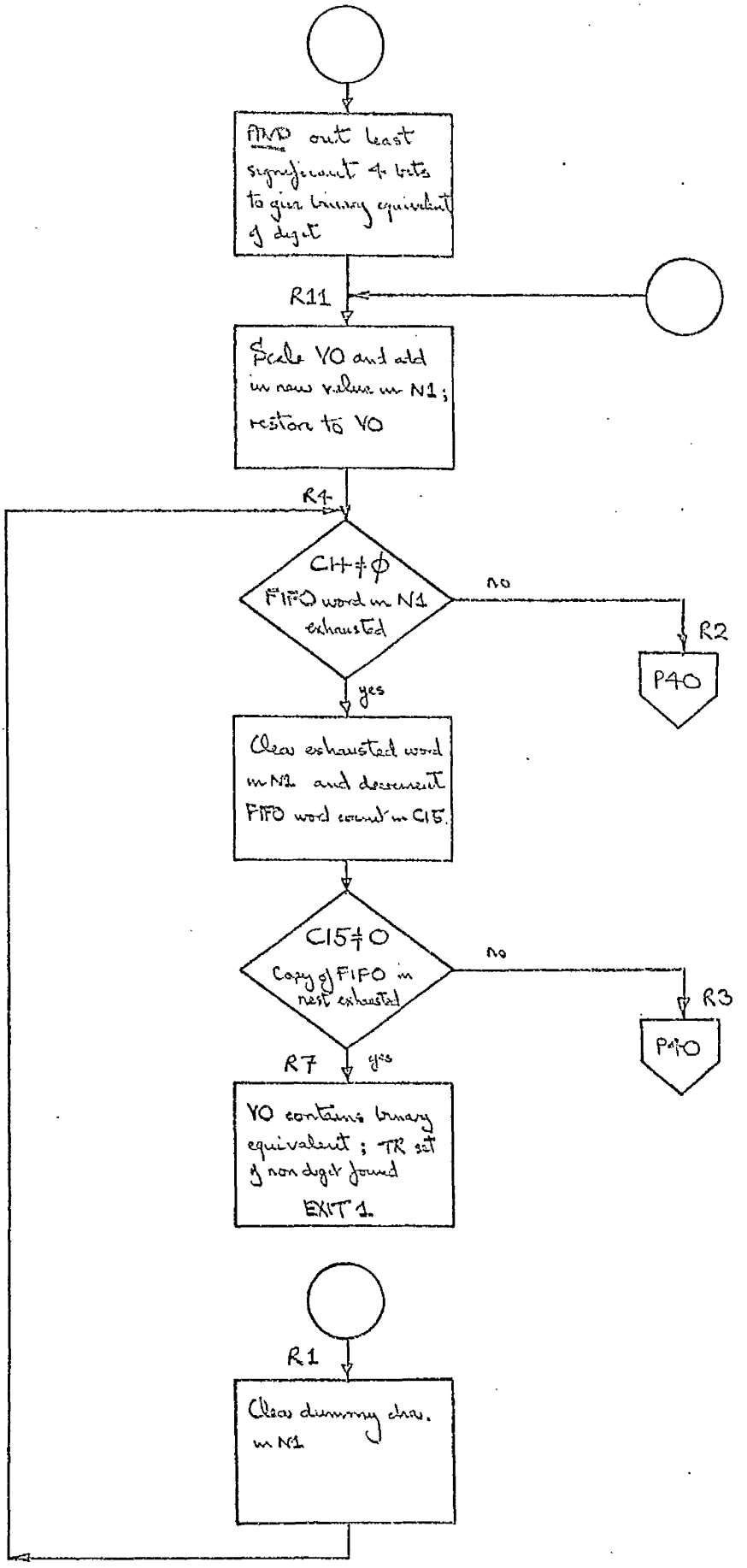

R2
Store N1 into next position in the gap in the sector buffer for this Terminal via M14M15@ ; PC7
C7 = ϕ
all next stored
no
yes
3 word message set up in sector buffer gap and OCW1, 2 and V34899.
EXIT 1

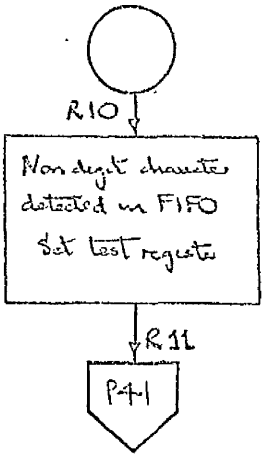

R3
JSP906
set M9 to terminal sector buffer b/a.
Check that it is permissible to recycle all prompts at this stage: E22 of sector buffer = ϕ.
E22 = ϕ
no
yes
Clear E11 of terminal sector buffer to restart prompt sequence with prompt zero.
EXIT 1

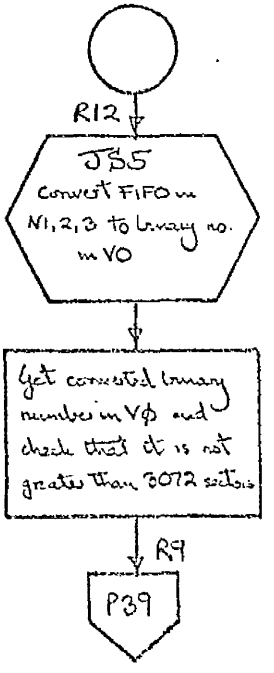

P812
Set terminal no. in N1 in C8
Fetch V3+P379 and shift by C8.
Change sign of N1, shift by C8 and restore I/O status altered
EXIT 1
P813
Clear assembly word V2 in P805.
Convert to characters
By shifting ensure number is only 6 x 8 bit characters — no warning if not
JS3 P805
insert parity bits
Fetch converted number from assembly word V2P805
EXIT 1

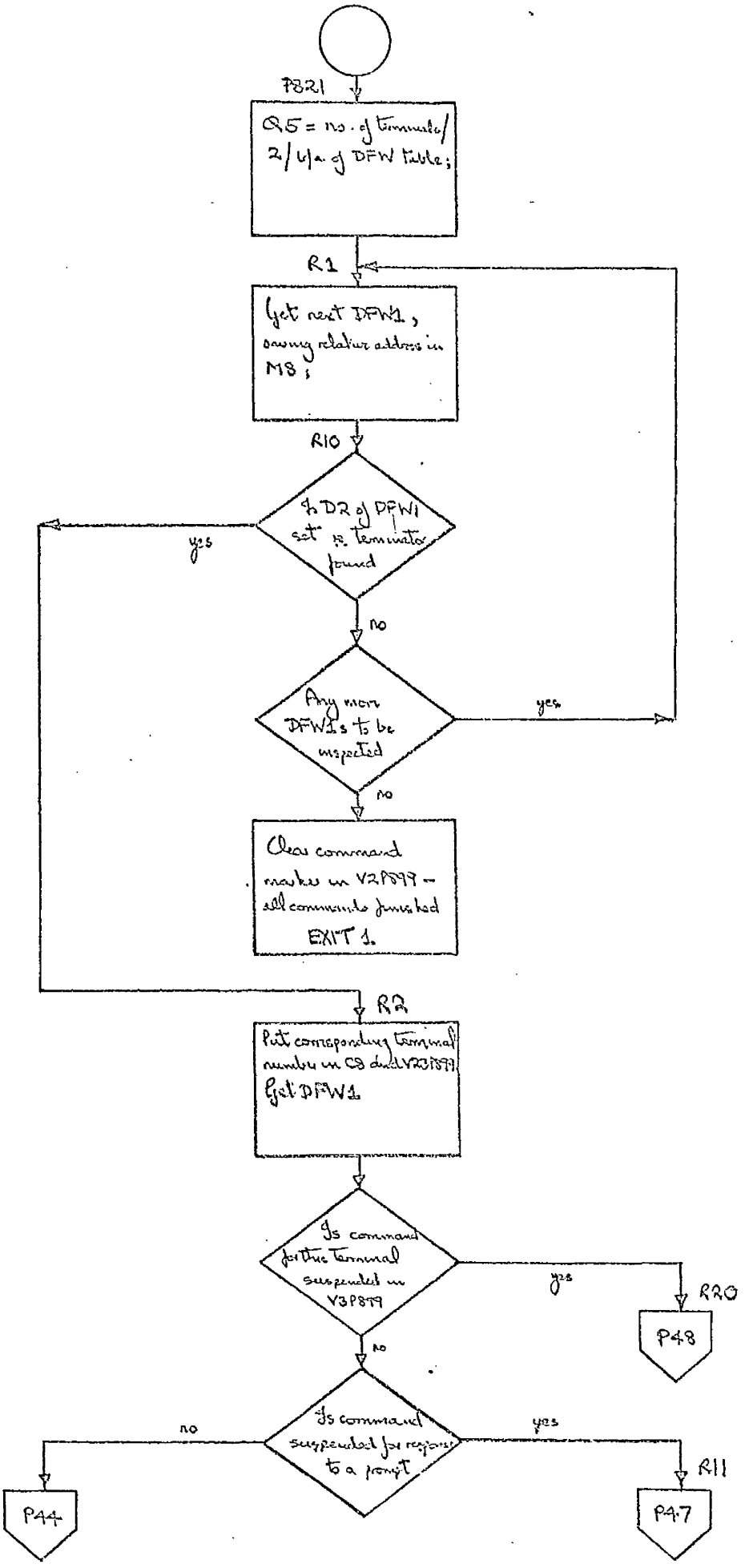

R6
P814
JS5
convert FIFO in N1,2,3 to binary number in N1 and store in V0
get converted binary number in V0 and check that it is not greate than 7200 secs.
R9
Clear VR ; check N2 less Than N1 ie. 0<N2<N1 in this case ; check response for range
N2 < N1 out of range
no
R7
P41
yes
TR set invalid character in R5
yes
R7
P41
no
FIFO contained a valid number within required range now in N1
EXIT2
R8
JS5
convert FIFO in N1,2,3 to binary number in V0.
get converted binary number in V0 and check that it is not greate than 14100 secs

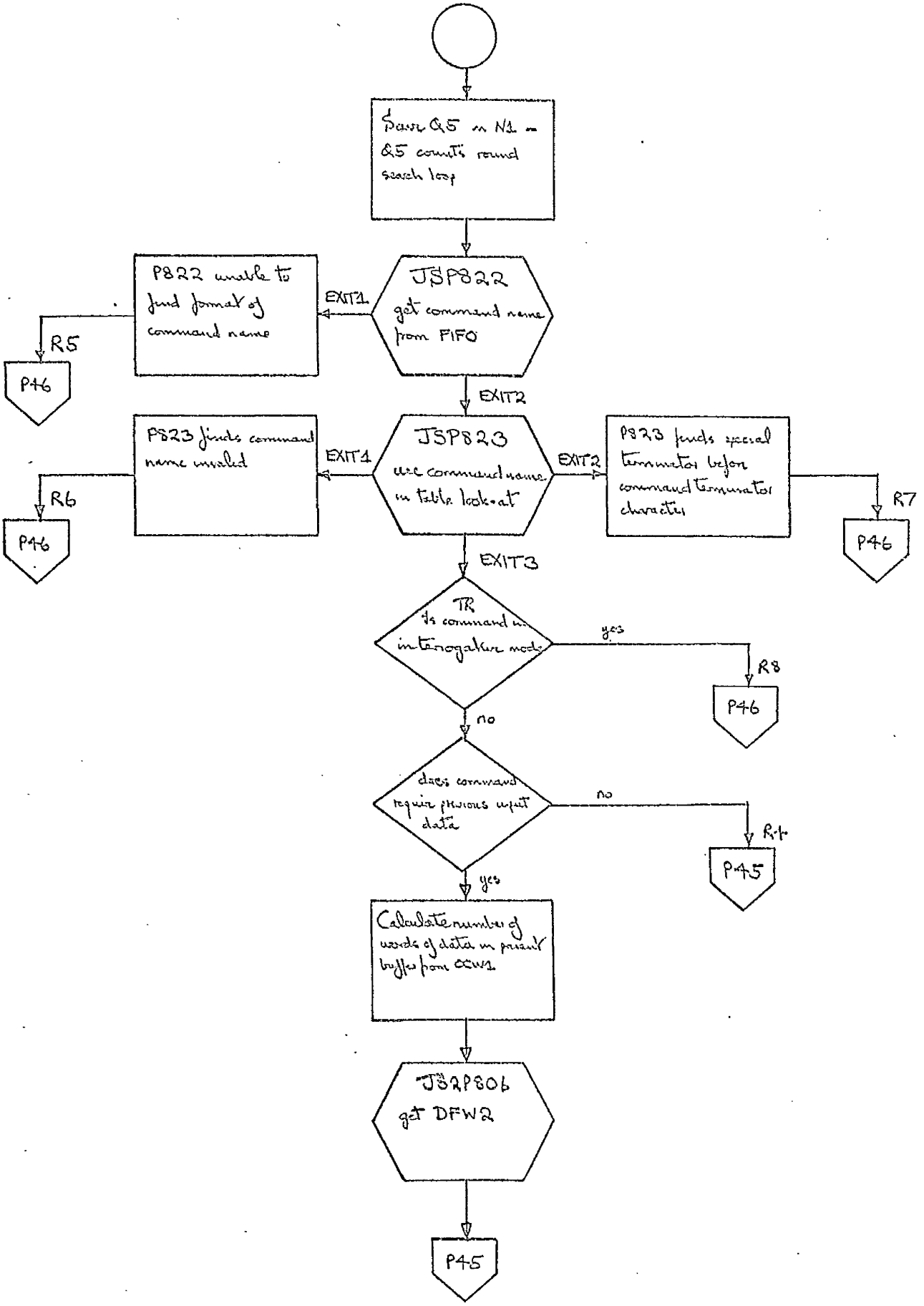

R5
Clear test registers; clear assembly word in VΦ; C15 = 3 to count FIFO format
R3
C14 = 6 to count chars from N1 (one word of FIFO)
R2
Spill out next character into N1 DC14
Is character in N1 = B377
Is character in N1 = BΦ
R1
P41
AND out least significant 7 bits from character in N1
Is character in N1 < B60
Is character in N1 > B71
R10
P42

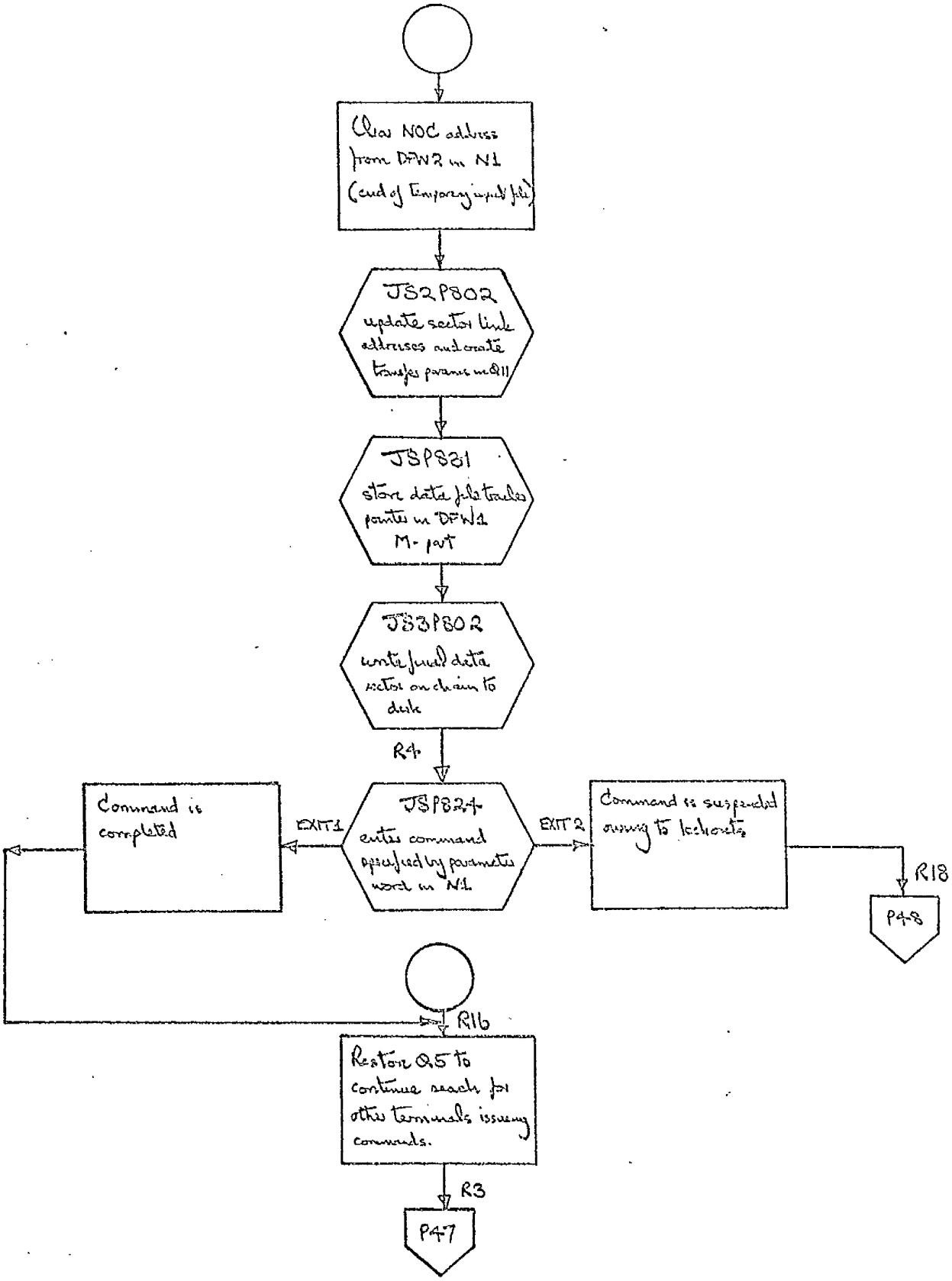

AND out least significant 4 bits to give binary equivalent of digit
R11
Scale Y0 and add in new value in N1; restore to Y0
R4
C14≠0 FIFO word in N1 exhausted
no
R2
P40
yes
Clear exhausted word in N1 and decrement FIFO word count in C15.
C15≠0 Copy of FIFO in next exhausted
no
R3
P40
R7
yes
Y0 contains binary equivalent ; TR set if non digit found
EXIT 1.
R1
Clear dummy char. in N1

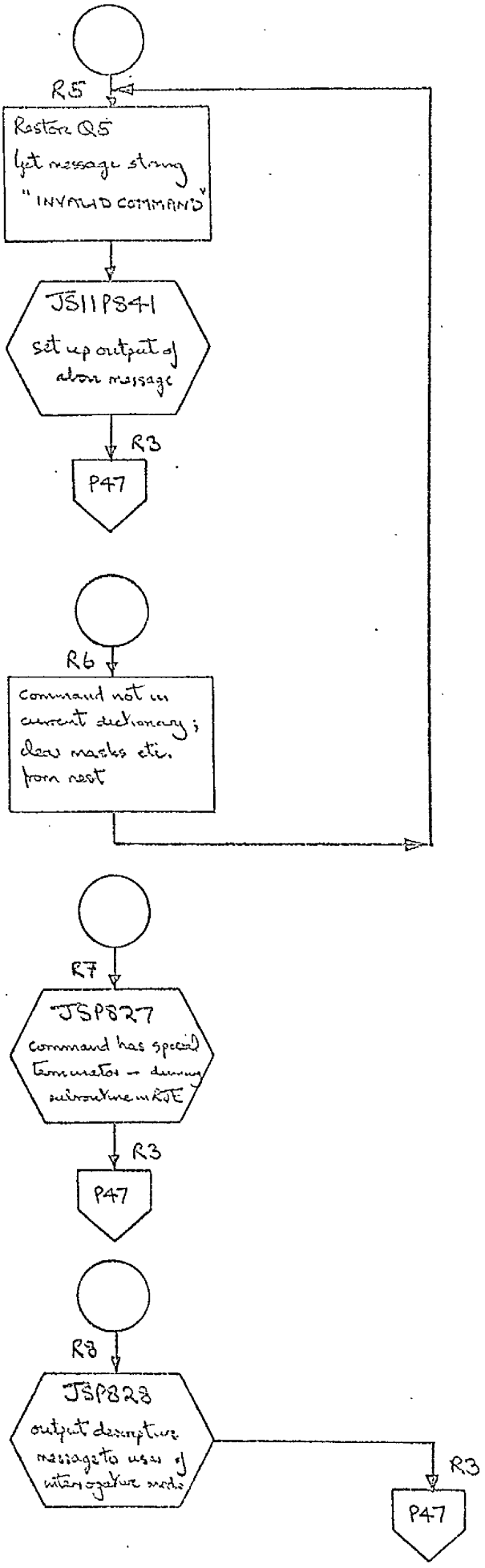

R10
Non digit character detected in FIFO
Set test register
R11
P41

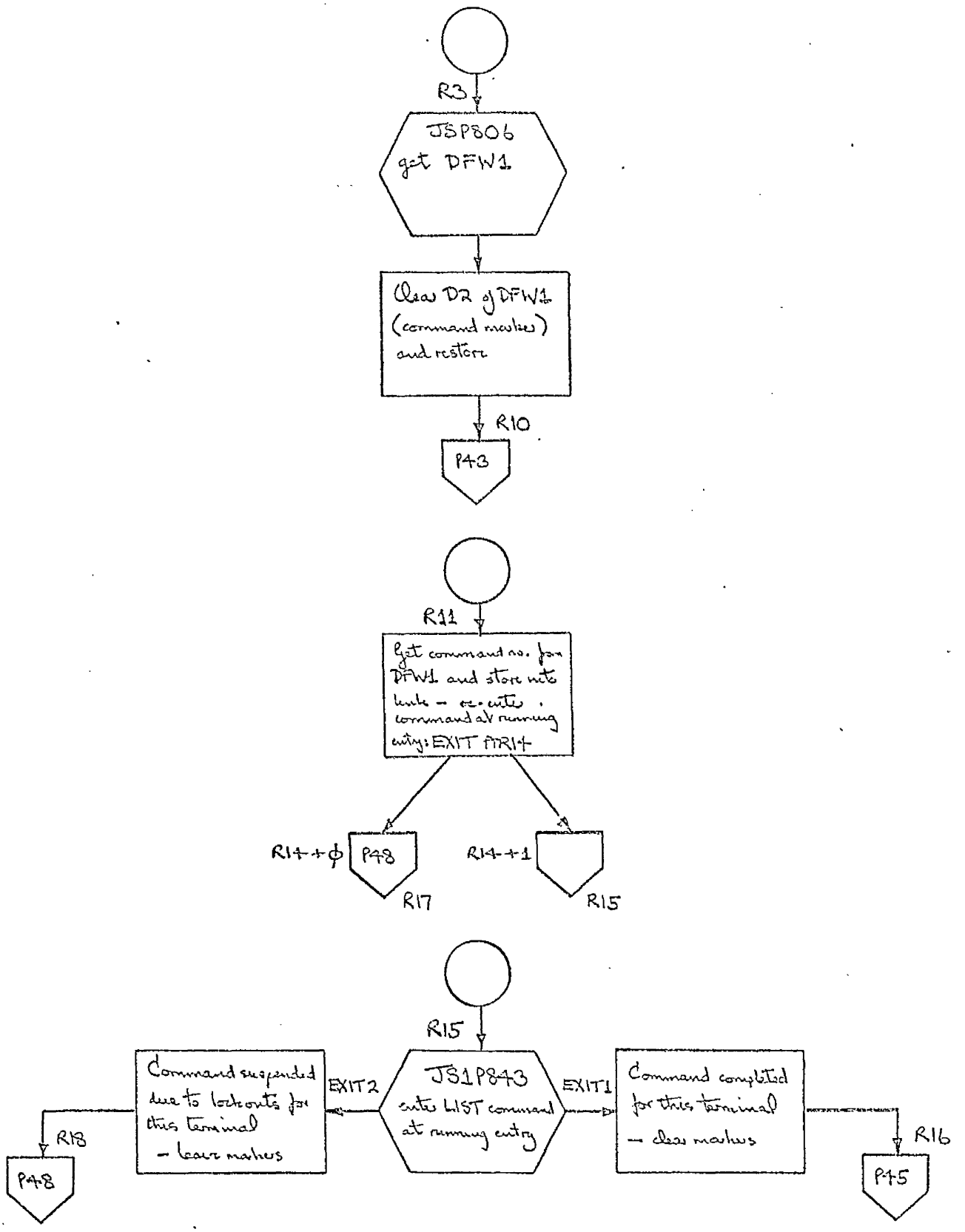

R12
JSS
Convert FIFO in N1,2,3 to binary no. in V0
Get converted binary number in V0 and check that it is not greater than 3072 octets
R9
P39

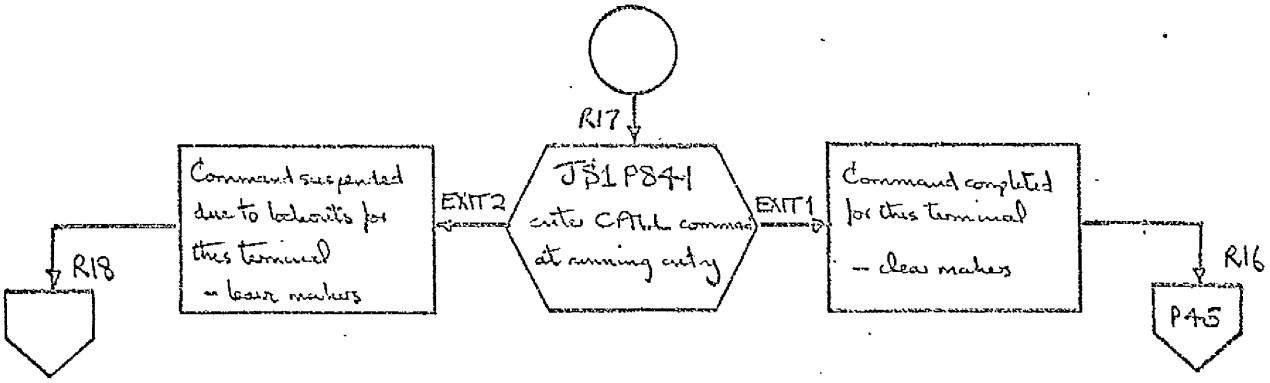

P821
Q5 = no. of terminals/2/4 a. of DFW table;
R1
Get next DFW1, saving relative address in M8;
R10
1st DR of DFW1 set to terminator found
yes
no
Any more DFW1s to be inspected
yes
no
Clear command marks in V2P899 – all commands finished EXIT 1.
R2
Put corresponding terminal number in CS and V2P899 Get DFW1
Is command for the terminal suspended in V3P899
yes
R20
P48
no
Is command suspended for response to a prompt
no
yes
R11
P44
P47

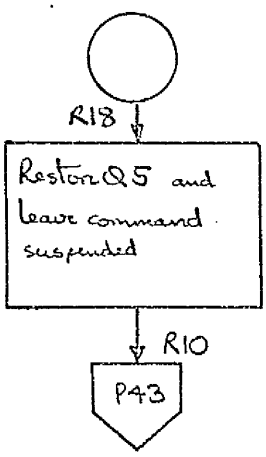

Save Q5 in N1 — Q5 counts round search loop
JSP822 get command name from FIFO
EXIT1
P822 unable to find format of command name
R5
P46
EXIT2
JSP823 use command name in table look-at
EXIT1
P823 finds command name invalid
R6
P46
EXIT2
P823 finds special terminator before command terminator character
R7
P46
EXIT3
TR is command in interrogative mode
yes
R8
P46
no
does command require previous input data
no
R+
P45
yes
Calculate number of words of data in present buffer from OCW2
JSP806 get DFW2
P45

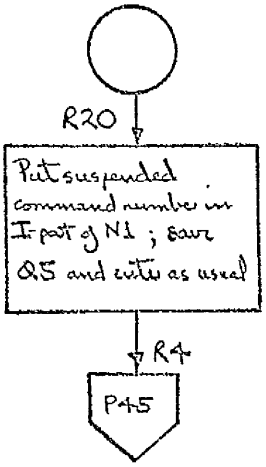

Clear NOC address from DFW2 in N1 (end of temporary input file)
JS2P802 update sector link addresses and create transfer parameter Q11
JSP821 store data file transfer pointers in DFW1 M-part
JS3P802 write final data sector on chain to disk
R4
JSP824 enters command specified by parameter word in N1
EXIT 1
Command is completed
EXIT 2
Command is suspended owing to lockout
R18
P48
R16
Restore Q5 to continue search for other terminals issuing commands.
R3
P47

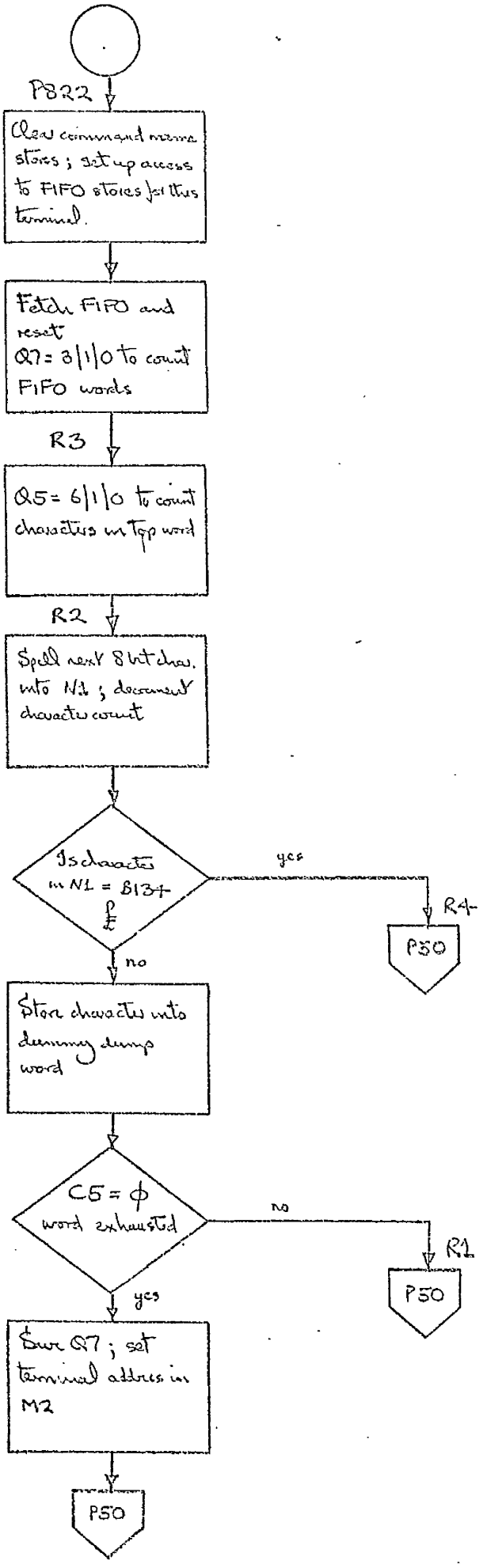

R5
Restore Q5
Get message string
"INVALID COMMAND"
JSIIPS41
set up output of
abov message
R3
P47
R6
Command not in
current dictionary;
clear marks etc.
from rest
R7
JSPS27
command has special
Terminator - dummy
subroutine in RTE
R3
P47
R8
JSPS28
output descriptive
message to user of
interrogative mode
R3
P47

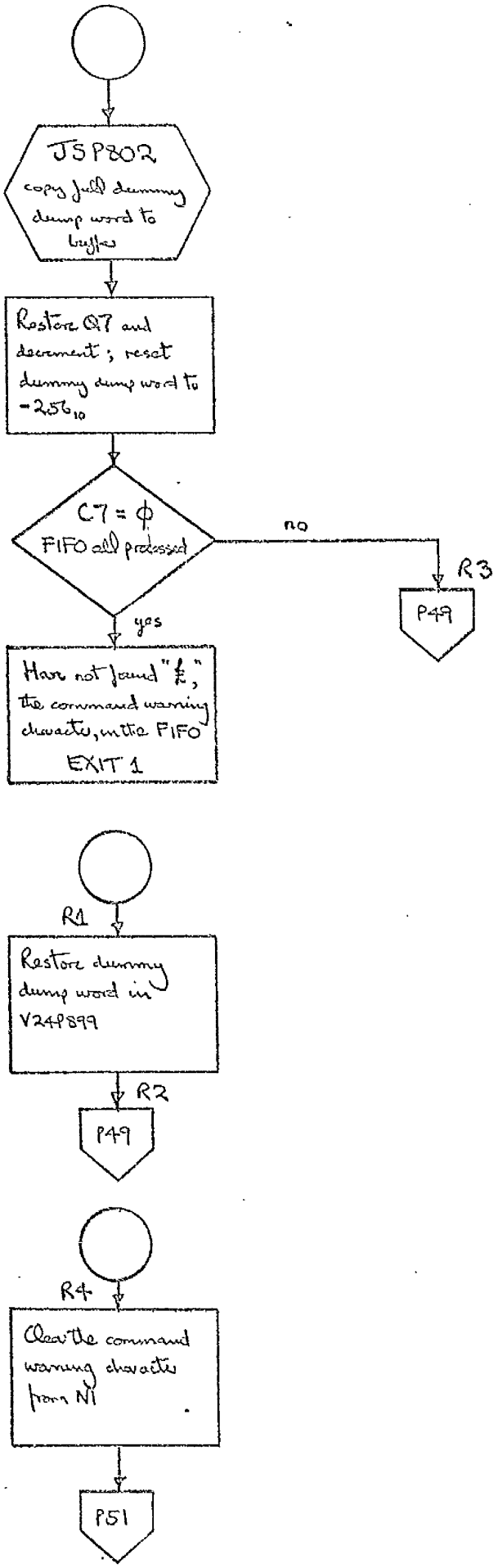

R3
JSP806
get DFW1
Clear D2 of DFW1
(command modes)
and restore
R10
P43
R11
Get command no. for
DFW1 and store into
link – execute
command at running
entry: EXIT from R14
R14++0
P48
R17
R14++1
R15
R15
JS1P843
enter LIST command
at running entry
EXIT2
Command suspended
due to lockouts for
this terminal
– leave modes
R18
P48
EXIT1
Command completed
for this terminal
– clear modes
R16
P45

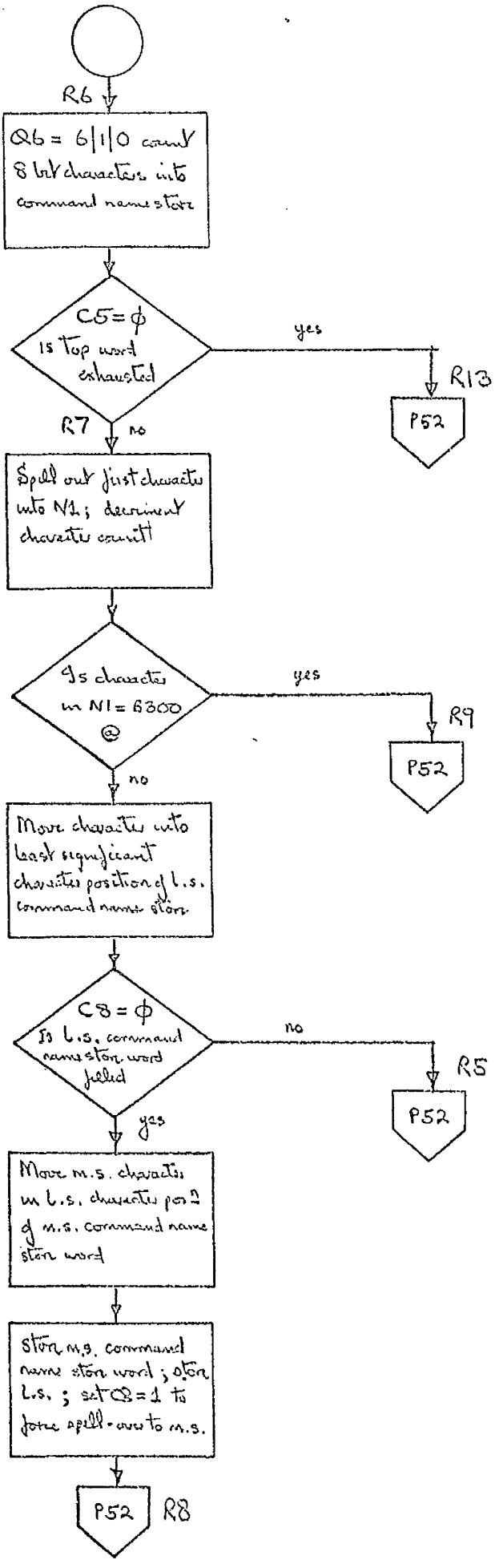

R17
JSI P841
enter CALL command
at running entry
EXIT 2
EXIT 1
Command suspended due to lockouts for this terminal
– leave markers
R18
Command completed for this terminal
– clear markers
R16
P45

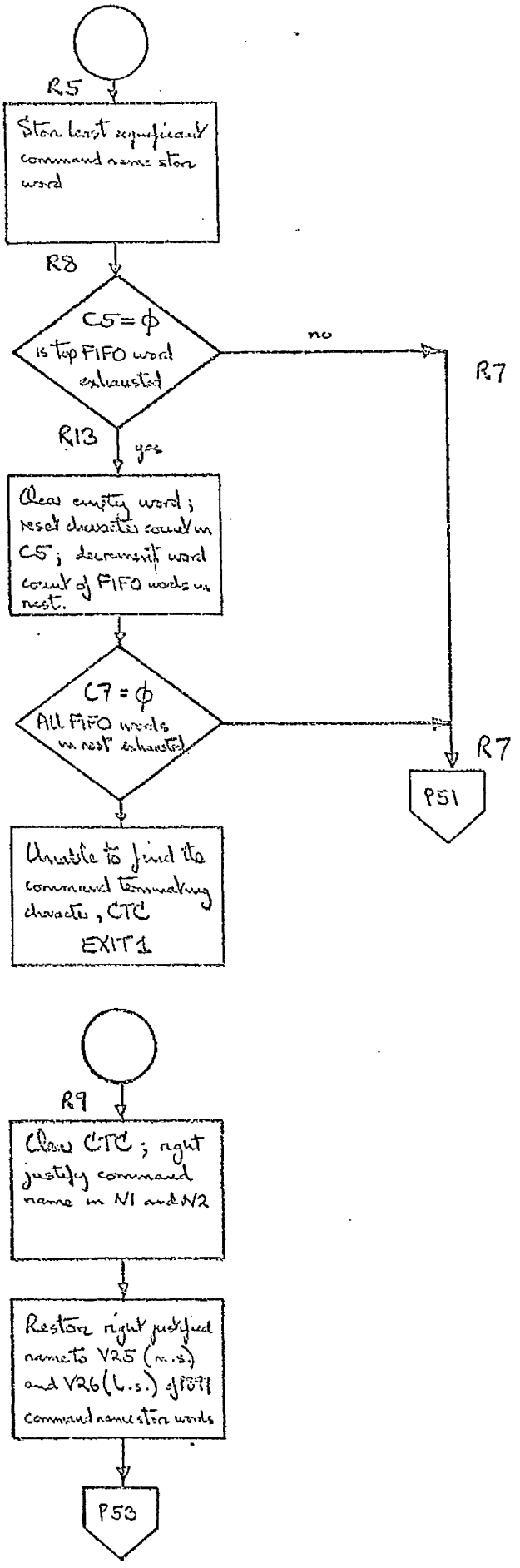

R18
Reset Q5 and leave command suspended
R10
P43

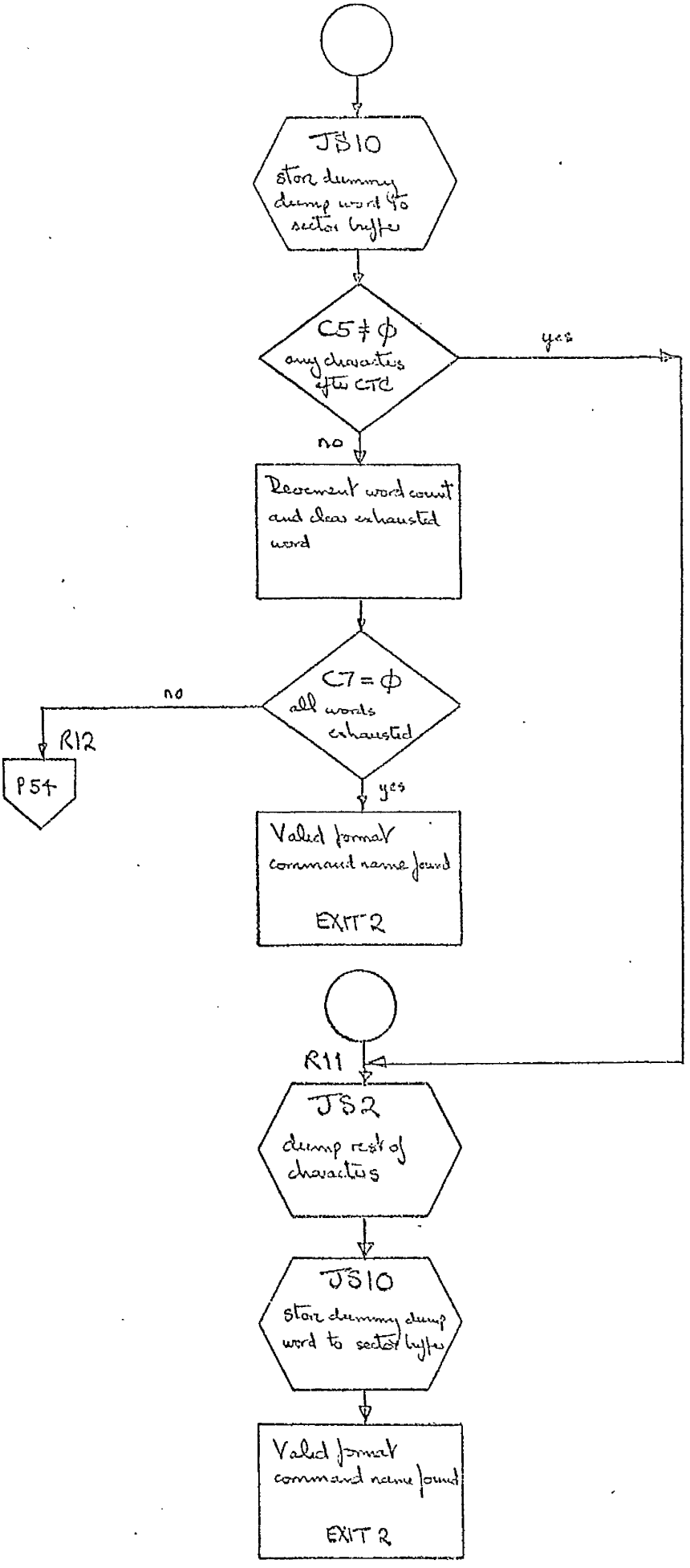

R20
Put suspended command number in I-part of N1 ; save Q5 and exit as usual
R4
P45

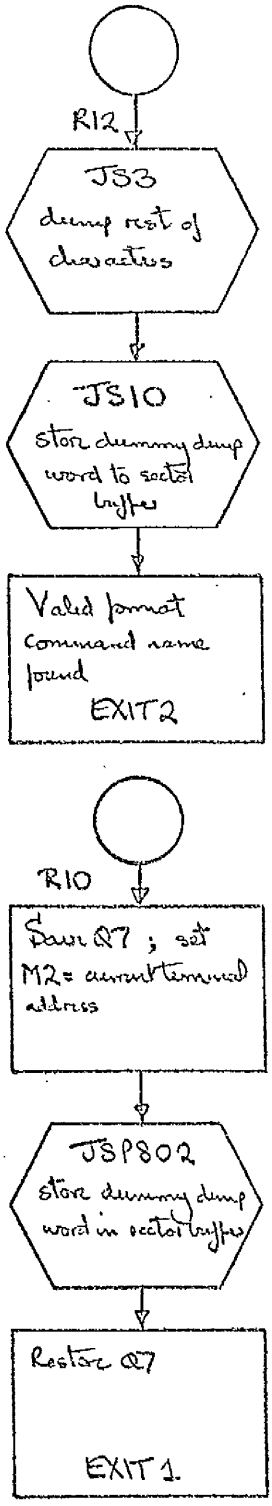

P822
Clear command mema stores ; set up access to FIFO stores for this terminal.
Fetch FIFO and reset Q7 = 3/1/0 to count FIFO words
R3
Q5 = 6/1/0 to count characters in Top word
R2
Spell next 8 bit char. into N1 ; decrement character count
Is character in N1 = B134 ?
yes
R4
P50
no
Store character into dummy dump word
C5 = ϕ word exhausted
no
R1
P50
yes
Swr Q7 ; set terminal address in M2
P50

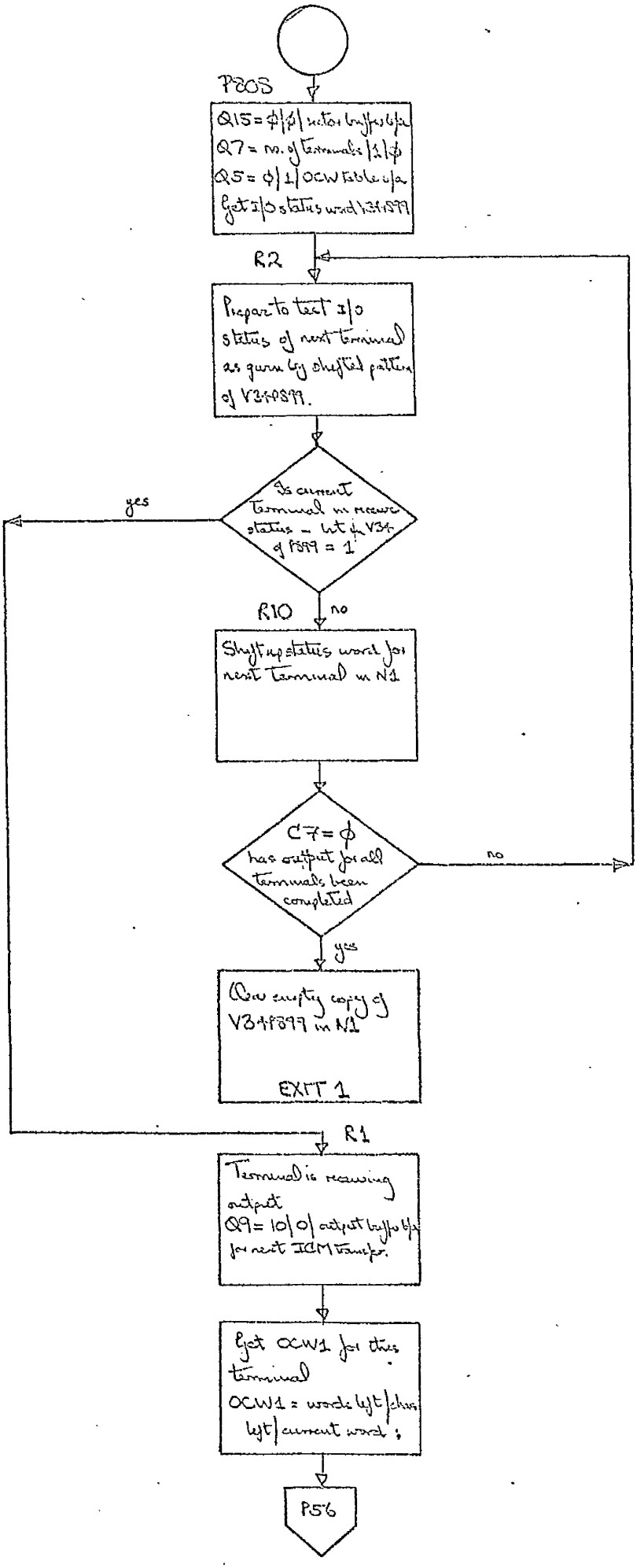

JSP802
copy full dummy dump word to buffer
Restore Q7 and decrement ; reset dummy dump word to -256_{10}
C7 = ϕ
FIFO all processed
no
R3
P49
yes
Have not found "£", the command warning character, in the FIFO
EXIT 1
R1
Restore dummy dump word in V24P899
R2
P49
R4
Clear the command warning character from N1
P51

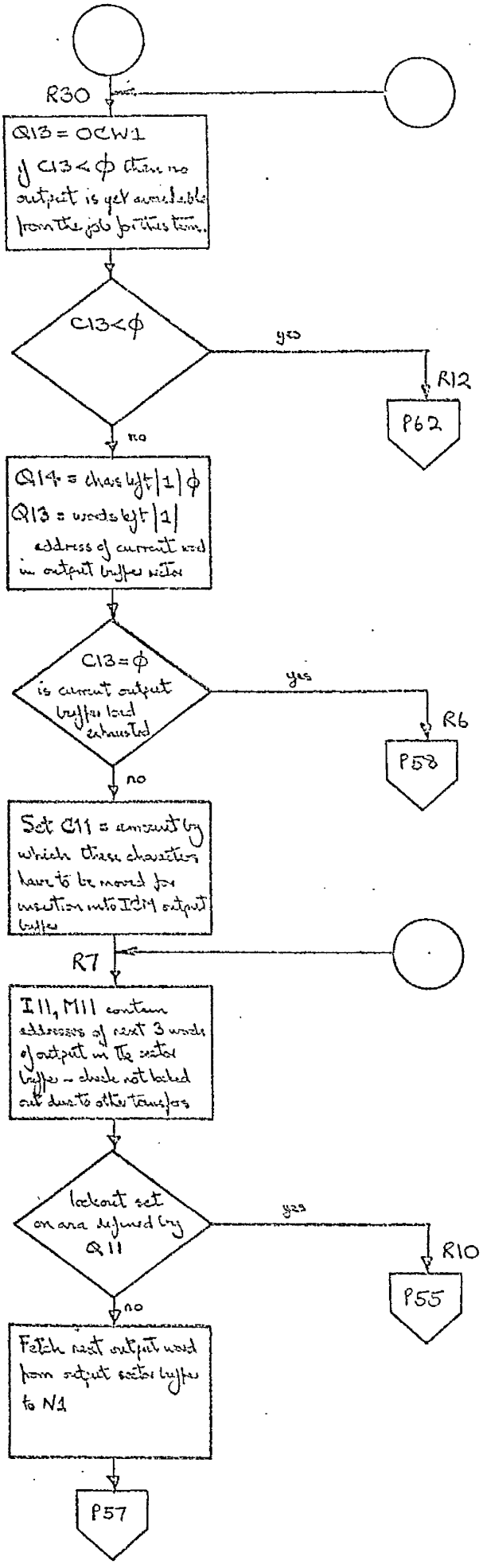

R6
Q6 = 6|1|0 count 8 bit characters into command name store
C5 = ϕ
Is Top word exhausted
yes
R13
P52
R7
no
Spill out first character into N1; decriment character count
Is character in N1 = 6300 @
yes
R9
P52
no
Move character into least significant character position of l.s. command name store
C8 = ϕ
Is l.s. command name store word filled
no
R5
P52
yes
Move m.s. character in l.s. character pos'n of m.s. command name store word
store m.s. command name store word; store l.s.; set C8 = 1 to force spill-over to m.s.
P52
R8

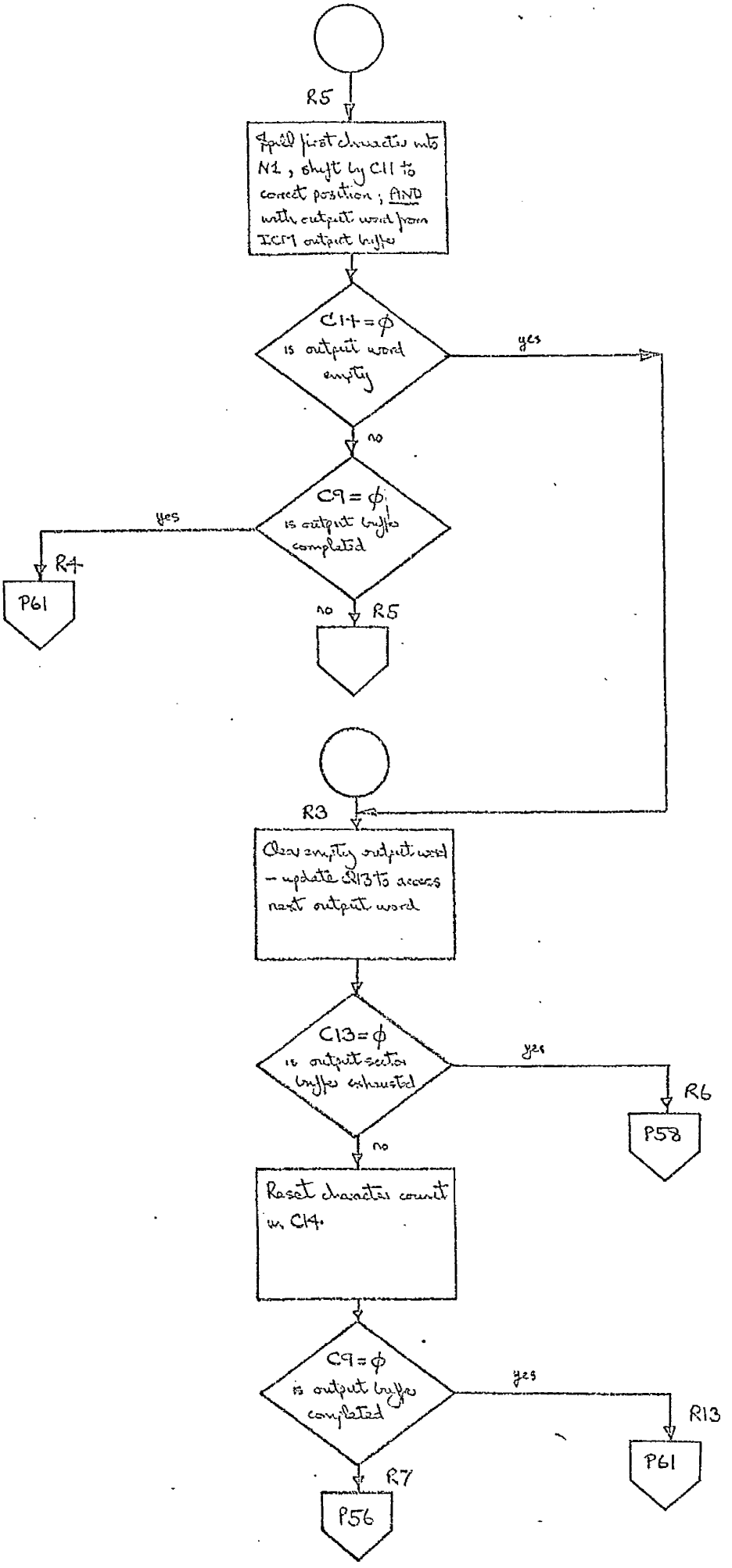

R5
Store least significant command name store word
R8
C5 = ϕ
Is top FIFO word exhausted
no
R7
R13
yes
Clear empty word; reset character count in C5; decrement word count of FIFO words in next.
C7 = ϕ
All FIFO words in next exhausted
R7
P51
Unable to find the command terminating character, CTC
EXIT 1
R9
Clear CTC; right justify command name in N1 and N2
Restore right justified name to V25 (m.s.) and V26 (l.s.) of FIFO command name store words
P53

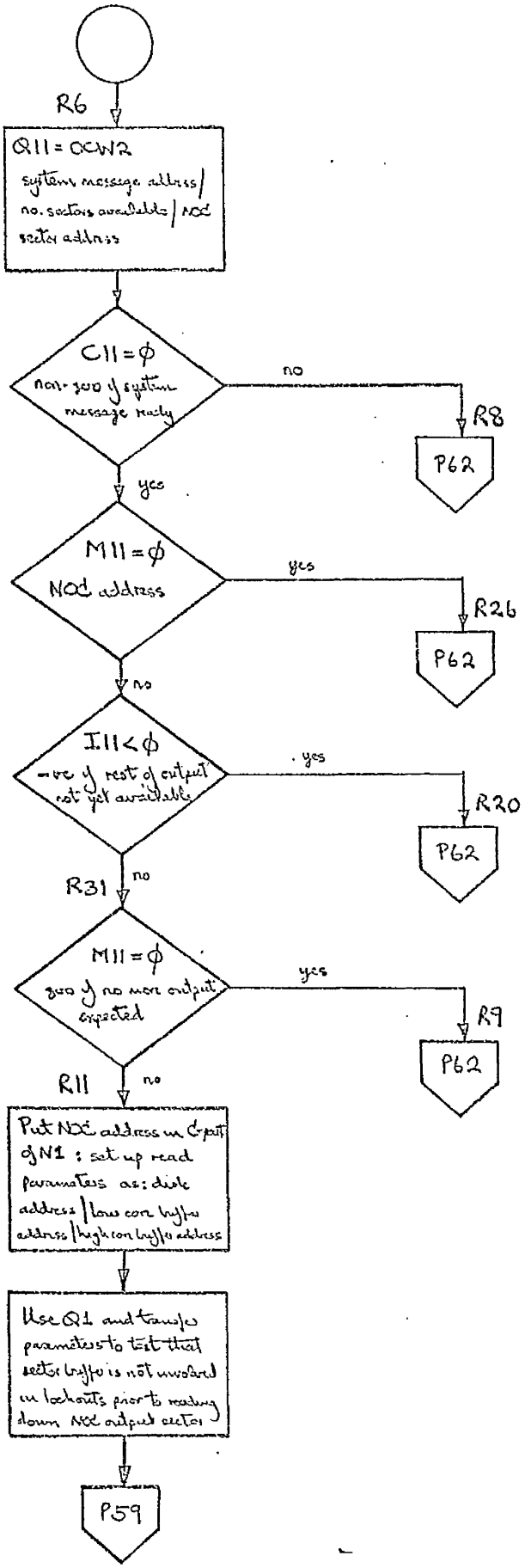

JS10
store dummy dump word to sector buffer
C5 ≠ ϕ
any characters after CTC
yes
no
Recement word count and clear exhausted word
C7 = ϕ
all words exhausted
no
R12
P54
yes
Valid format command name found
EXIT 2
R11
JS2
dump rest of characters
JS10
store dummy dump word to sector buffer
Valid format command name found
EXIT 2

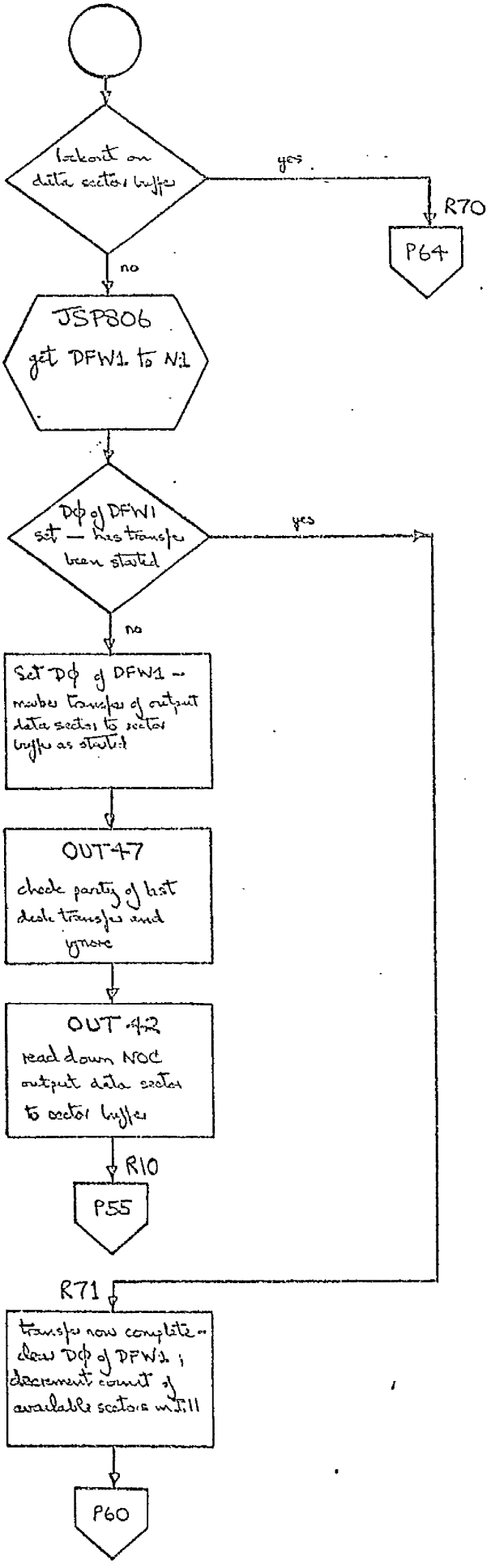

R12
JS3
dump rest of characters
JS10
store dummy dump word to sector buffer
Valid format command name found
EXIT 2
R10
Save Q7 ; set M2 = current terminal address
JSP802
store dummy dump word in sector buffer
Restore Q7
EXIT 1.

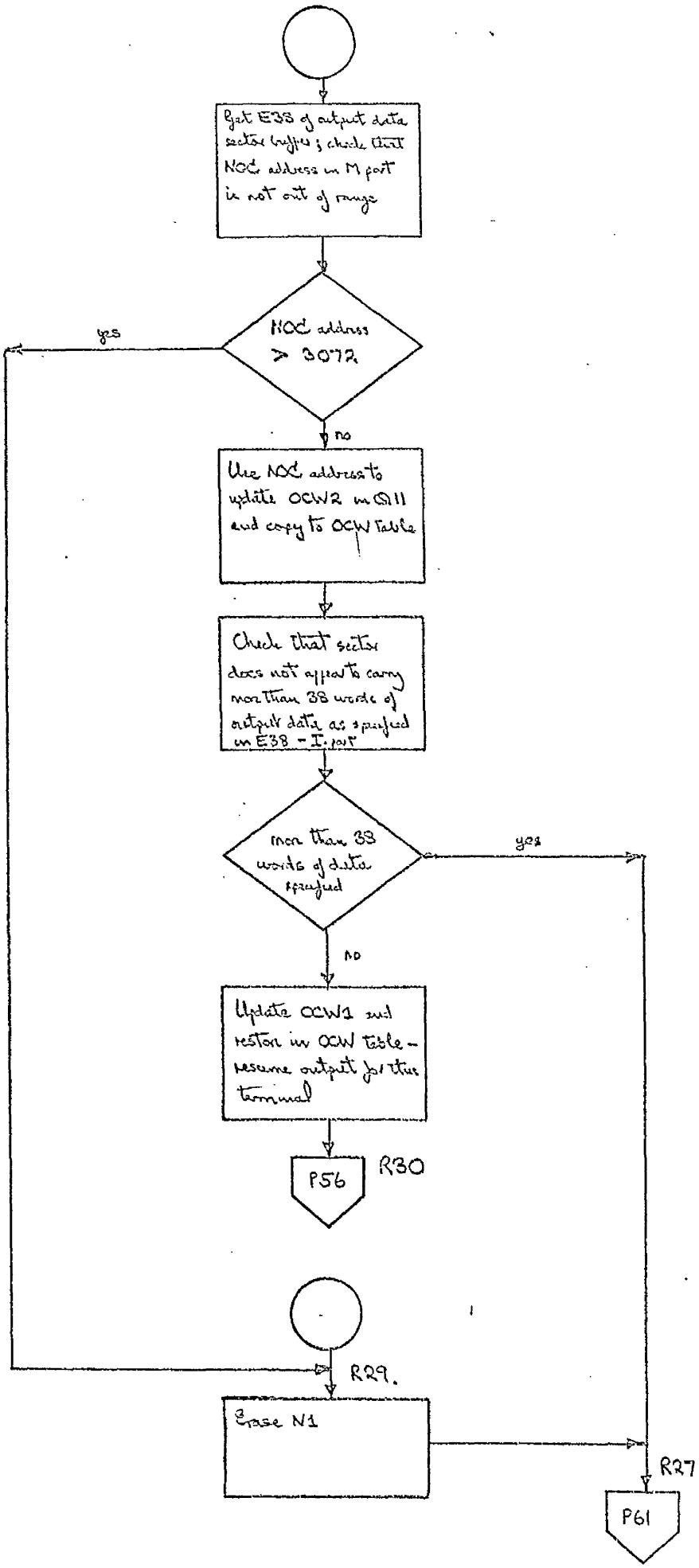

P80S
Q15 = ϕ/1/ vector buffer o/p
Q7 = no. of terminals/1/ϕ
Q5 = ϕ/1/ OCW table o/p
Get I/O status word V3+899
R2
Prepare to test I/O status of next terminal as given by shifted pattern of V3+899.
Is current Terminal in receive status - bit ϕ V3+ of P899 = 1
yes
no
R10
Shift up status word for next Terminal in N1
C7 = ϕ has output for all terminals been completed
no
yes
Clear empty copy of V3+P899 in N1
EXIT 1
R1
Terminal is receiving output
Q9 = 10/0/ output buffer o/p for next ICM transfer.
Get OCW1 for this terminal
OCW1 = words left/chars left/current word ;
P56

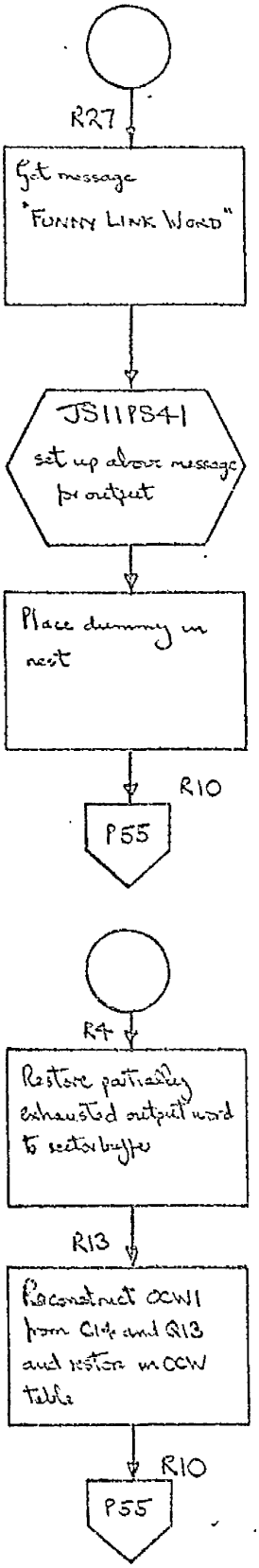

R30
Q13 = OCW1
if C13 < ϕ then no output is yet available from the job for this term.
C13<ϕ
yes
R12
P62
no
Q14 = chars left | 1 | ϕ
Q13 = words left | 1 |
address of current word in output buffer sector
C13 = ϕ
is current output buffer load exhausted
yes
R6
P58
no
Set C11 = amount by which these characters have to be moved for insertion into IDM output buffer
R7
I11, M11 contain addresses of next 3 words of output in the sector buffer - these not locked out due to other transfers
lockout set on area defined by Q11
yes
R10
P55
no
Fetch next output word from output sector buffer to N1
P57

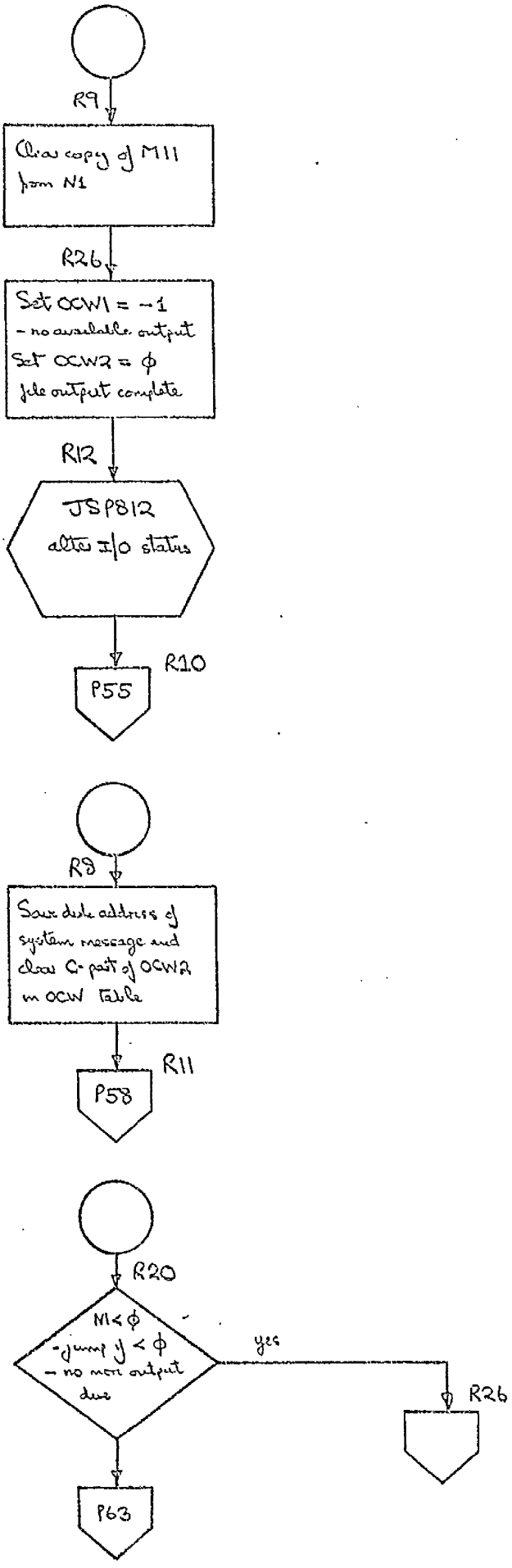

R5
Spill first character into N1, shift by C11 to correct position; AND with output word from ICM output buffer
C14 = ϕ
is output word empty
yes
no
C9 = ϕ
is output buffer completed
yes
R4
P61
no
R5
R3
Clear empty output word - update C13 to access next output word
C13 = ϕ
is output sector buffer exhausted
yes
R6
P58
no
Reset character count in C14.
C9 = ϕ
is output buffer completed
yes
R13
P61
R7
P56

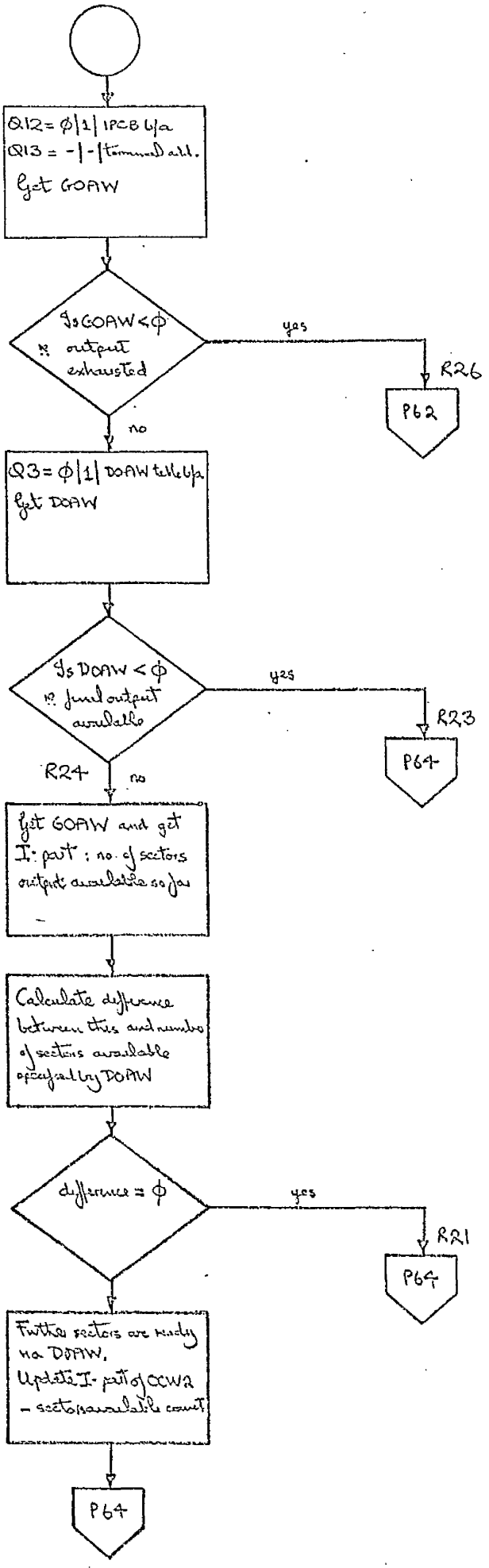

R6
Q11 = OCW2
system message address / no. sectors available / NOC sector address
C11 = ϕ
non-zero if system message ready
no
R8
P62
yes
M11 = ϕ
NOC address
yes
R26
P62
no
I.11 < ϕ
-ve if rest of output not yet available
yes
R20
P62
R31
no
M11 = ϕ
zero if no more output expected
yes
R9
P62
R11
no
Put NOC address in O/put of N1 : set up read parameters as: disk address / low core buffer address / high core buffer address
Use Q1 and transfer parameters to test that sector buffer is not unused in lockouts prior to reading down NOC output sector
P59

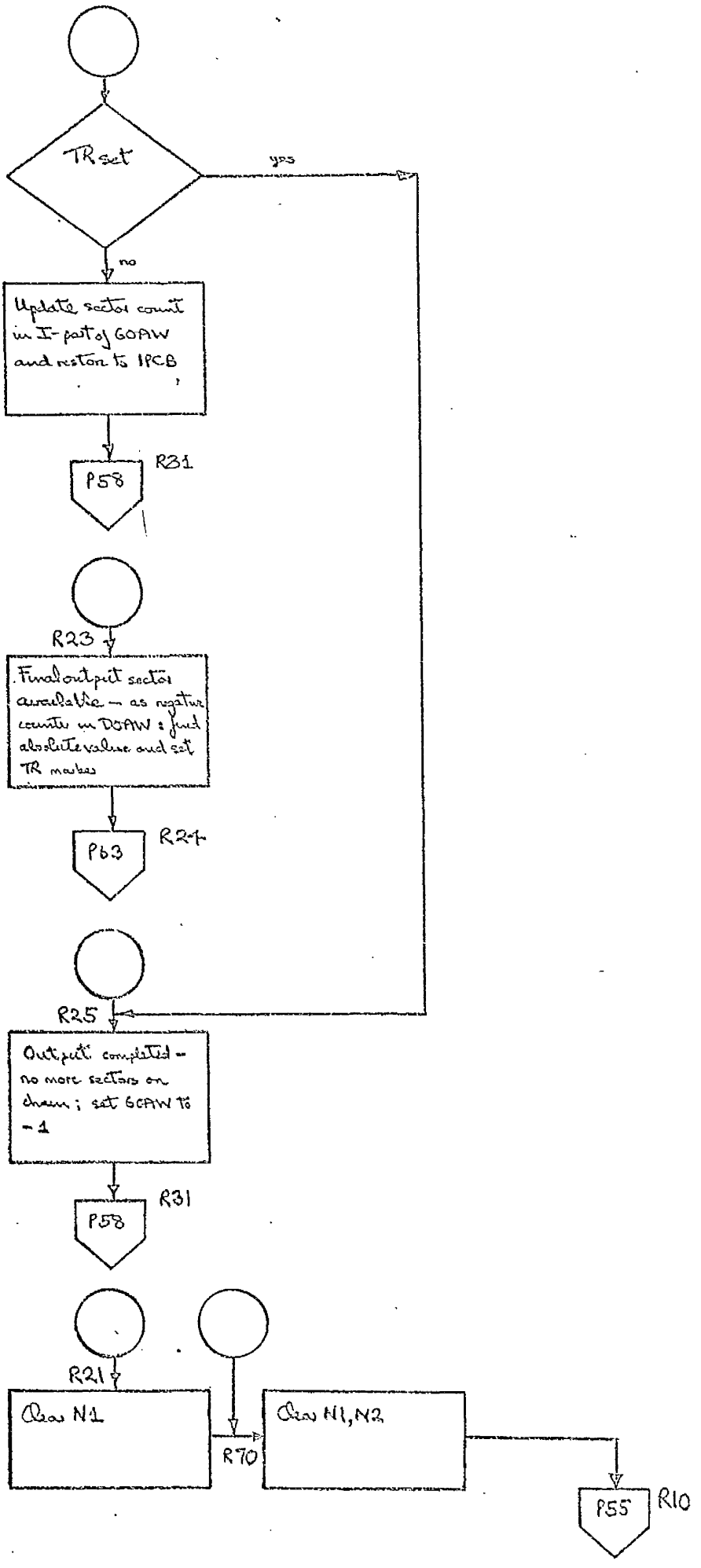

lockout on data sector buffer
yes
R70
P64
no
JSP806
get DFW1 to N:1
Dϕ of DFW1 set — has transfer been started
yes
no
Set Dϕ of DFW1 — number transfers of output data sectors to sector buffer as started
OUT47
check parity of last disk transfer and ignore
OUT42
read down NOC output data sector to sector buffer
R10
P55
R71
transfer now complete — clear Dϕ of DFW1; decrement count of available sectors until
P60

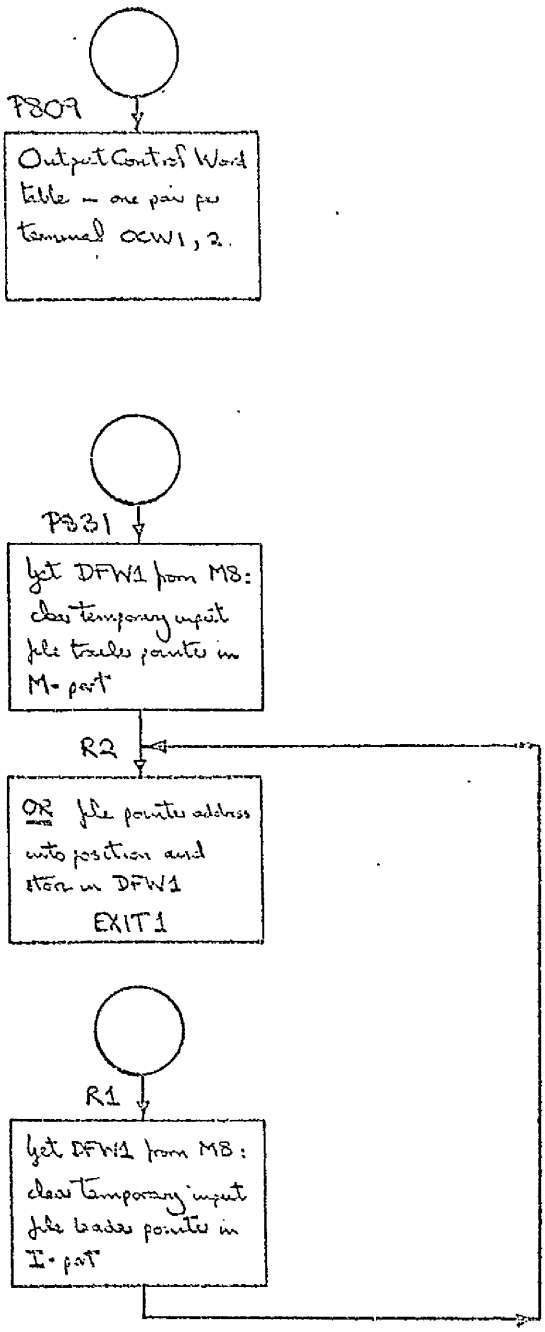

Get E38 of output data sector buffer; check that NOC address in M port is not out of range
NOC address > 3072.
yes
no
Use NOC address to update OCW2 in CB11 and copy to OCW table
Check that sector does not appear to carry more than 38 words of output data as specified in E38 - I. port
more than 38 words of data specified
yes
no
Update OCW1 and restore in OCW table - resume output for the terminal
P56
R30
R29.
Erase N1
R27
P61

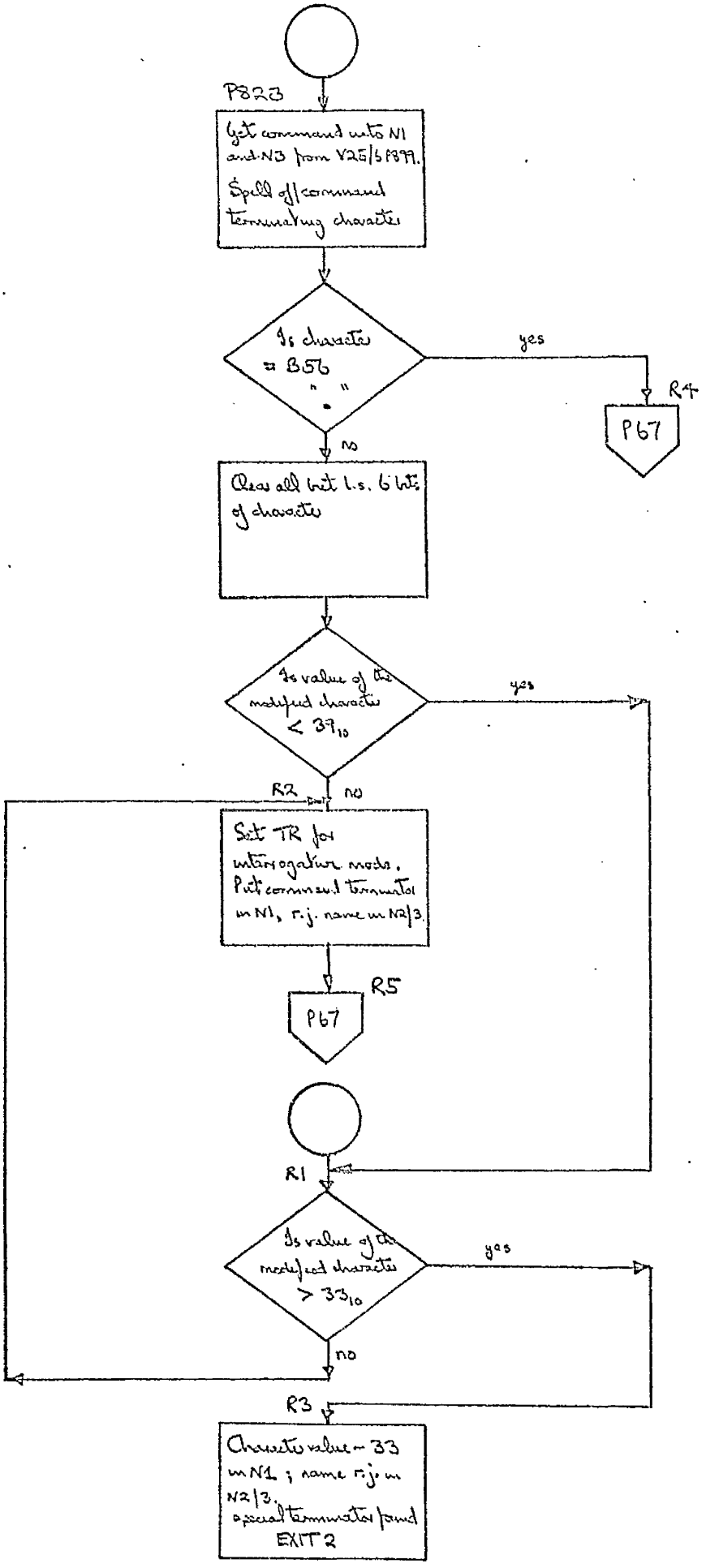

R27
Get message "FUNNY LINK WORD"
JS11PS41
set up above message for output
Place dummy in next
R10
P55
R4
Restore partially exhausted output word to sector buffer
R13
Reconstruct CCW1 from C14 and R13 and restor in CCW table
R10
P55

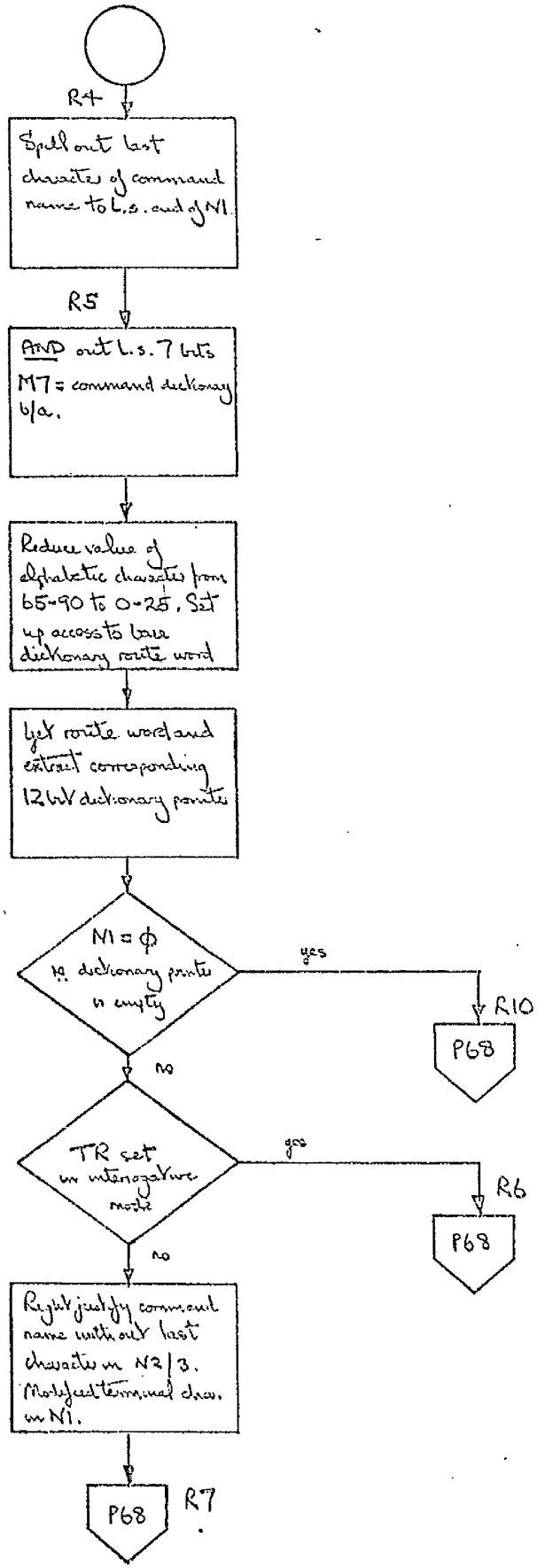

R9
Clear copy of M11 from N1
R26
Set OCW1 = -1 - no available output
Set OCW2 = ϕ file output complete
R12
JSP812 alter I/O status
R10
P55
R8
Save disk address of system message and clear C-part of OCW2 in OCW Table
R11
P58
R20
M1 < ϕ - jump if < ϕ - no more output due
yes
R26
P63

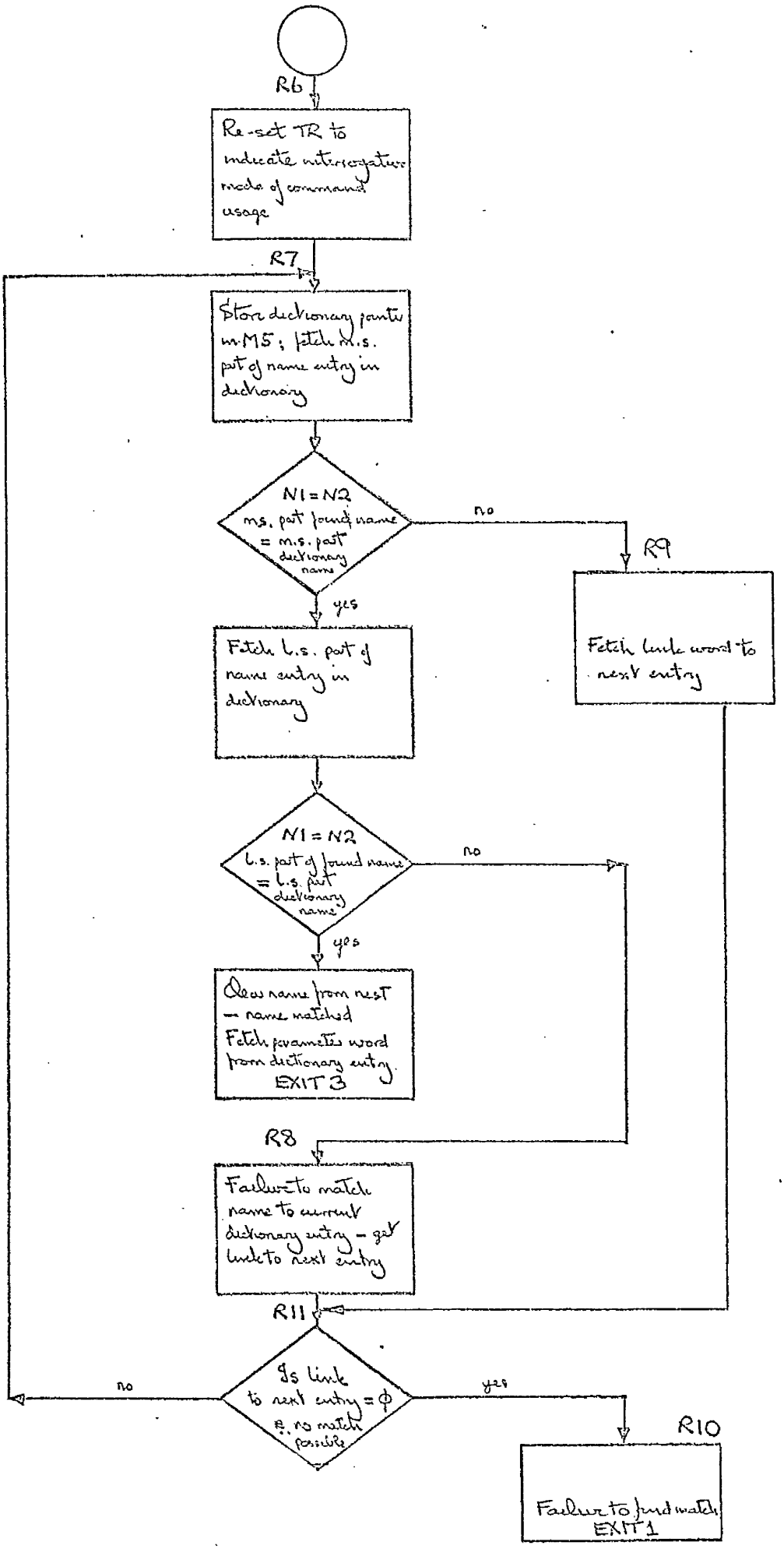

Q12 = ϕ|1| IPCB b/a
Q13 = -|-|terminal all.
Get GOAW
Is GOAW < ϕ
No output exhausted
yes
R26
P62
no
Q3 = ϕ|1| DOAW table b/a
Get DOAW
Is DOAW < ϕ
No fund output available
yes
R23
P64
R24
no
Get GOAW and get I- part : no. of sectors output available so far
Calculate difference between this and numbe of sectors available specified by DOAW
difference = ϕ
yes
R21
P64
Further sectors are ready na DOAW.
Update I- part of OCW2 - sectors available count
P64

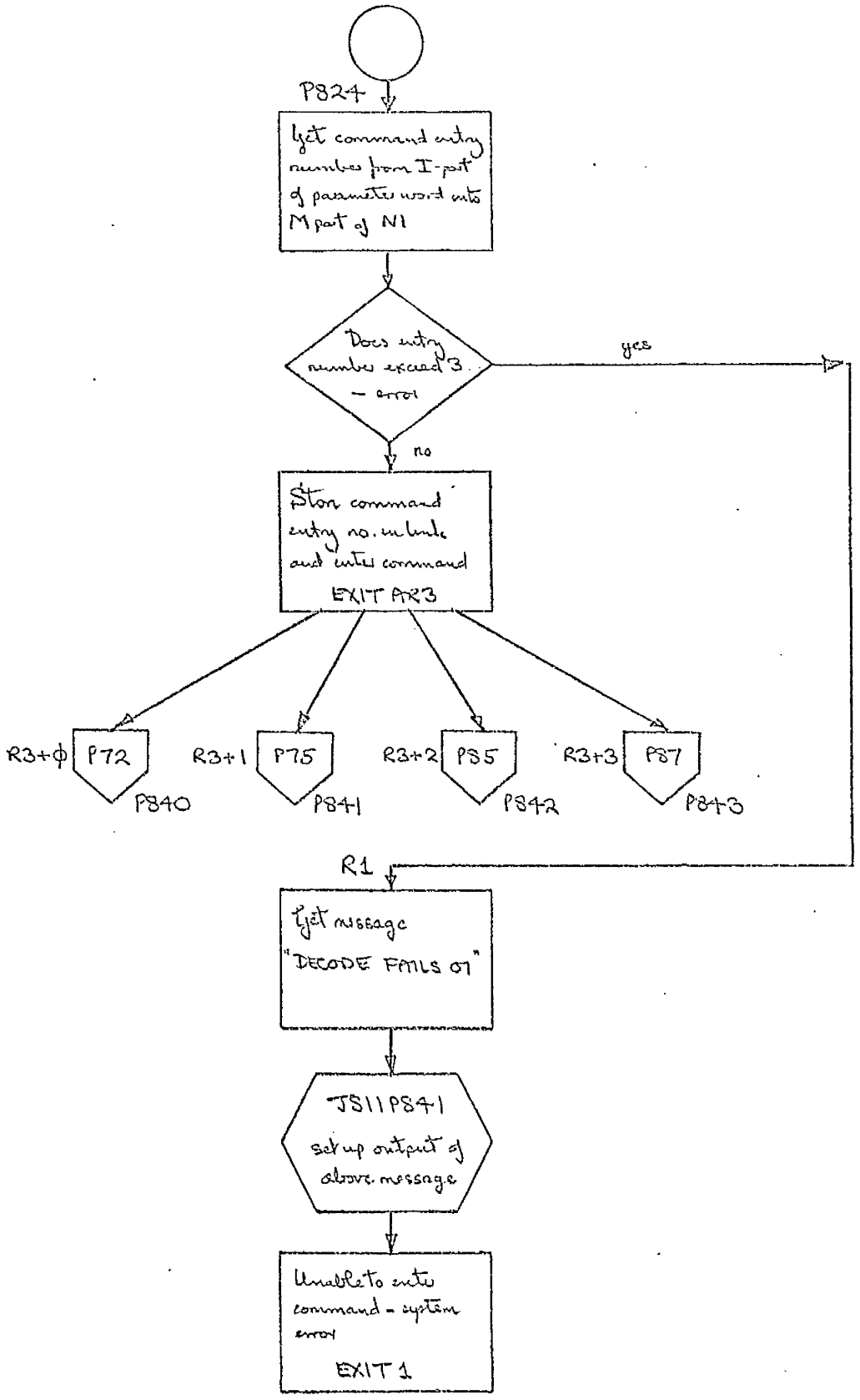

TR set
yes
no
Update sector count in I-part of GOAW and restore to IPCB
P58
R31
R23
Final output sector available — as negative counter in DOAW : find absolute value and set TR marker
P63
R24
R25
Output completed — no more sectors on chain ; set GOAW to -1
P58
R31
R21
Chan N1
R70
Chan N1, N2
P55
R10

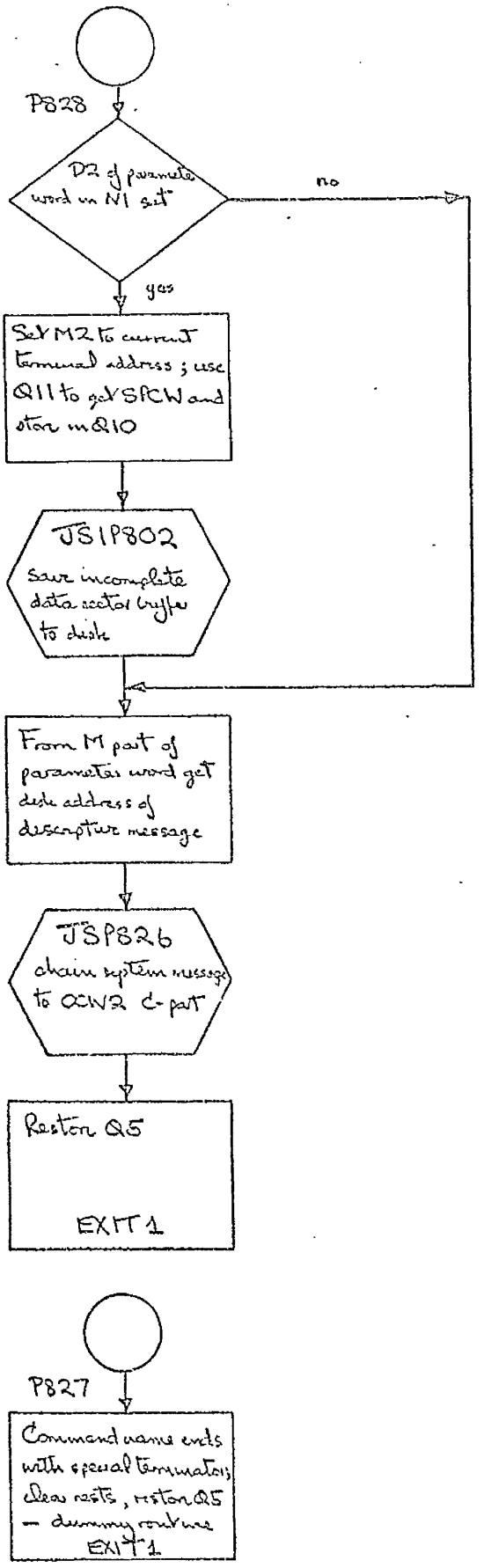

P809
Output Control Word table – one pair per terminal OCW1, 2.
P831
Get DFW1 from MB: clear temporary input file header pointer in M-part
R2
OR file pointer address into position and store in DFW1
EXIT 1
R1
Get DFW1 from MB: clear temporary input file header pointer in I-part

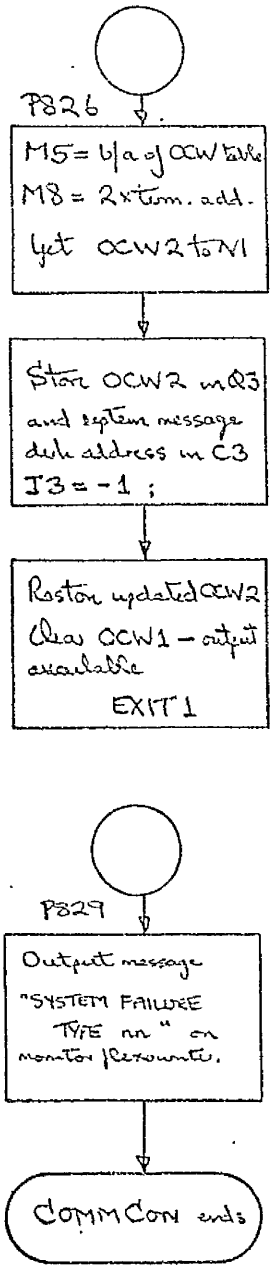

P823
Get command into N1 and N3 from V25/6 P899.
Spill off command terminating character
Is character = B56 " " "
yes
R4
P67
no
Clear all but l.s. 6 bits of character
Is value of the modified character < 37₁₀
yes
R2
no
Set TR for interrogation mode. Put command terminator in N1, r.j. name in N2/3.
R5
P67
R1
Is value of the modified character > 33₁₀
yes
no
R3
Character value = 33 in N1 ; name r.j. in N2/3. special terminator found
EXIT 2

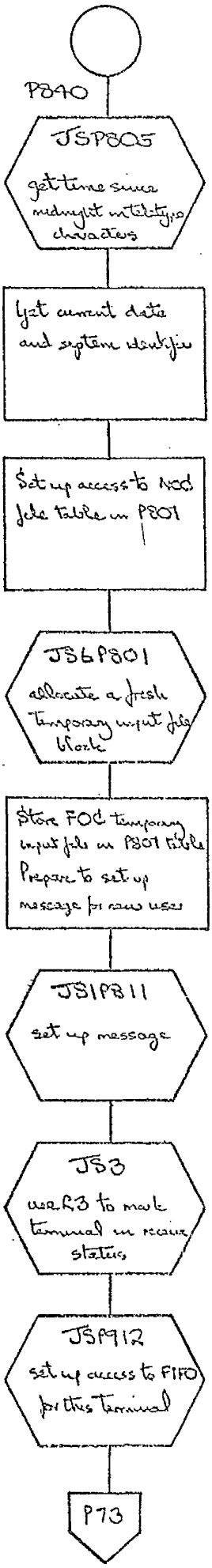

R4
Spell out last character of command name to L.s. end of N1.
R5
AND out L.s. 7 bits M7 = command dictionary b/a.
Reduce value of alphabetic characters from 65-90 to 0-25. Set up access to base dictionary route word
Get route word and extract corresponding 12bit dictionary pointer
N1 = ϕ ie. dictionary pointer is empty
yes
R10
P68
no
TR set in interrogative mode
yes
R6
P68
no
Right justify command name without last character in N2/3. Modified terminal char. in N1.
P68
R7

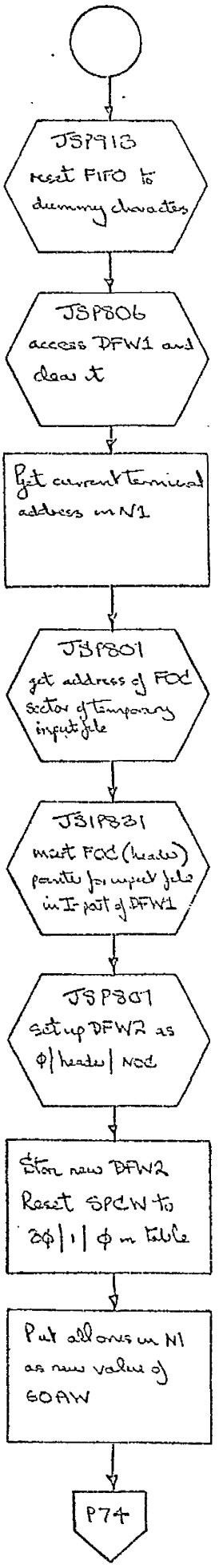

R6
Re-set TR to indicate interrogative mode of command usage
R7
Store dictionary pointer in M5; fetch M.S. part of name entry in dictionary
N1 = N2 m.s. part found name = m.s. part dictionary name
no
yes
R9
Fetch link word to next entry
Fetch L.S. part of name entry in dictionary
N1 = N2 L.S. part of found name = L.S. part dictionary name
no
yes
Clear name from rest — name matched Fetch parameter word from dictionary entry EXIT 3
R8
Failure to match name to current dictionary entry — get link to next entry
R11
Is link to next entry = ϕ i.e. no match possible
no
yes
R10
Failure to find match EXIT 1

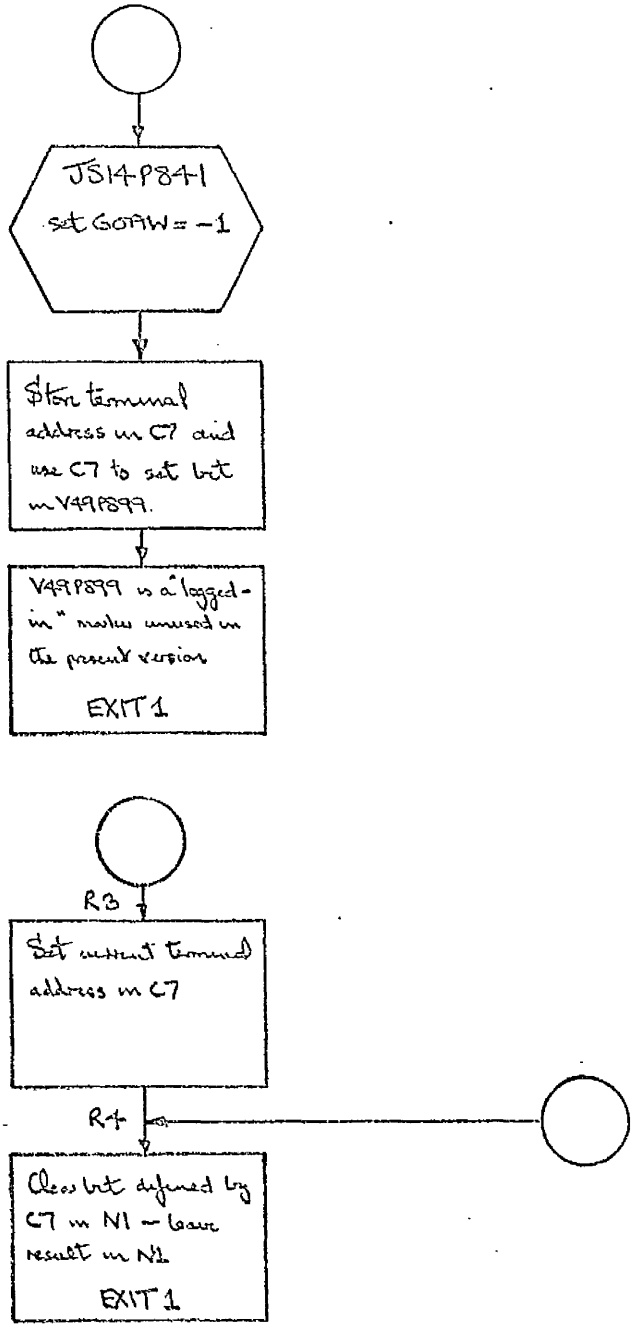

P824
Get command entry number from I-part of parameter word into M part of N1
Does entry number exceed 3 — error
yes
no
Store command entry no. in link and enter command
EXIT AR3
R3+0
P72
P840
R3+1
P75
P841
R3+2
P85
P842
R3+3
P87
P843
R1
Get message "DECODE FAILS 01"
JS11P841
set up output of above message
Unable to enter command - system error
EXIT 1

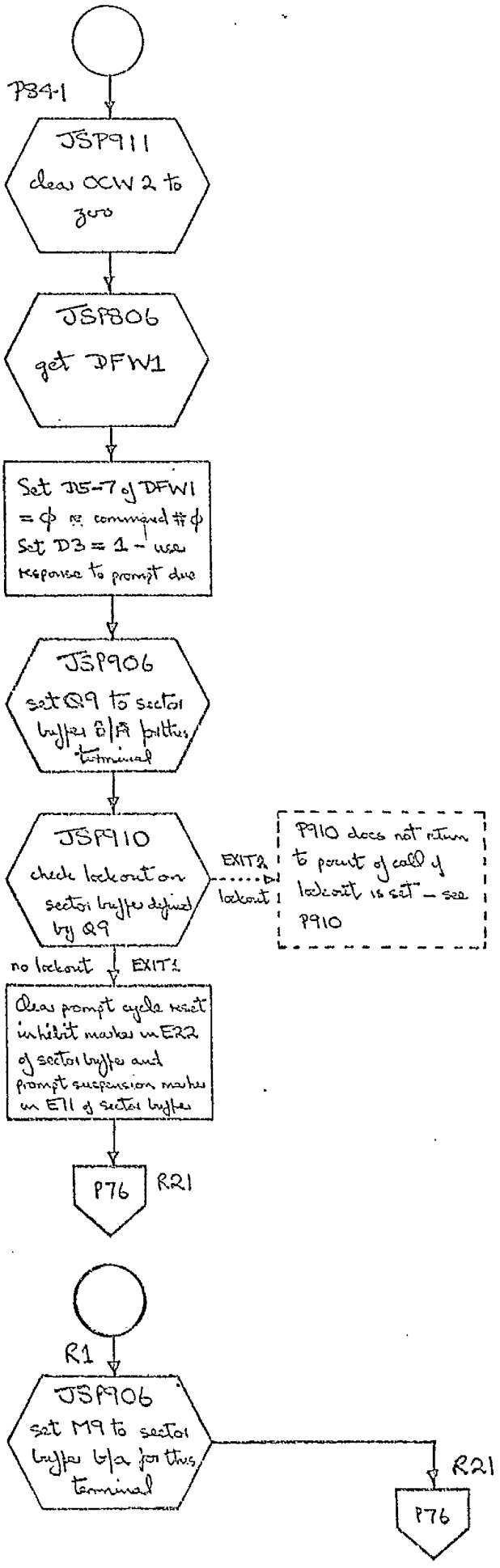

P828
D2 of parmete word on N1 set
no
yes
Set M2 to current terminal address ; use Q11 to get VSFCW and stor in Q10
JSP802
Save incomplete data secter buffer to disk
From M part of parameter word get disk address of descriptor message
JSP826
chain system message to QCN2 C-part
Reston Q5
EXIT 1
P827
Command name ends with special terminator; clear rests , reston Q5 — dummy routine
EXIT 1

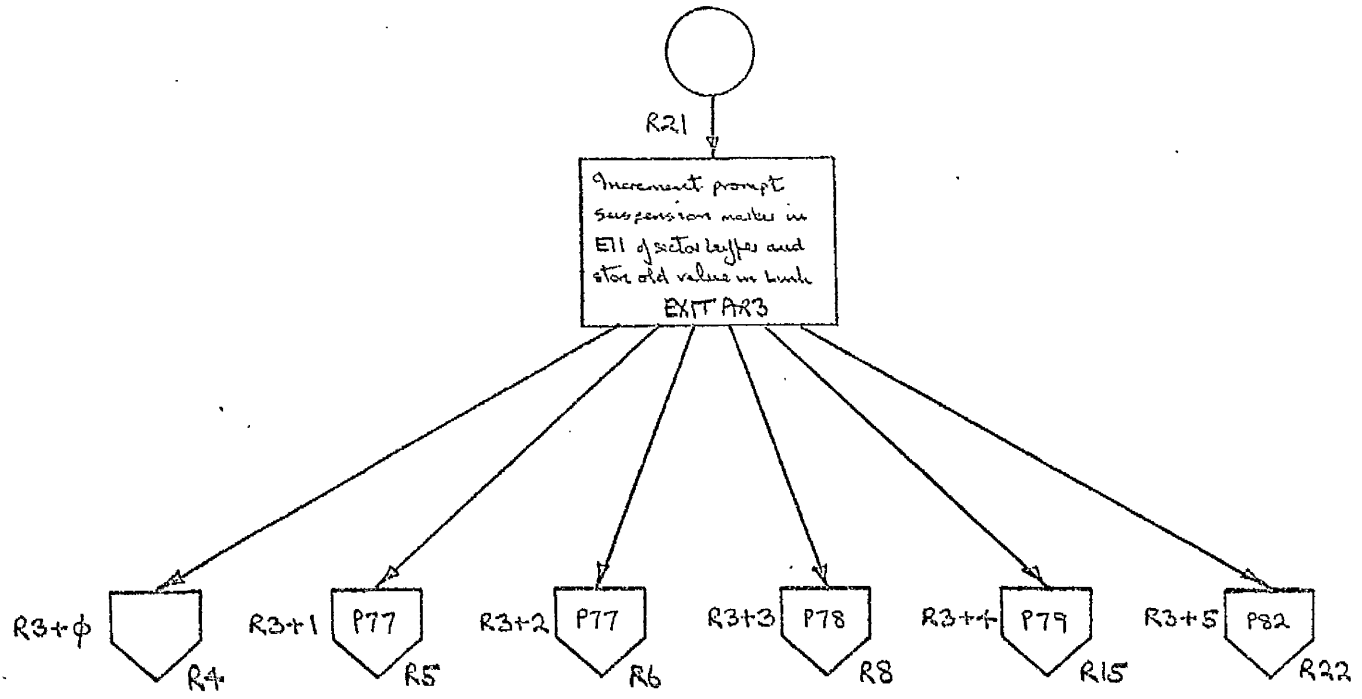

P826
M5 = b/a of OCW table
M8 = 2×term. add.
Get OCW2 to M1
Store OCW2 in Q3 and system message disk address in C3
J3 = -1 ;
Restore updated OCW2
Clear OCW1 – output available
EXIT 1
P829
Output message
"SYSTEM FAILURE
TYPE nn " on monitor/keyboard.
COMMCON ends

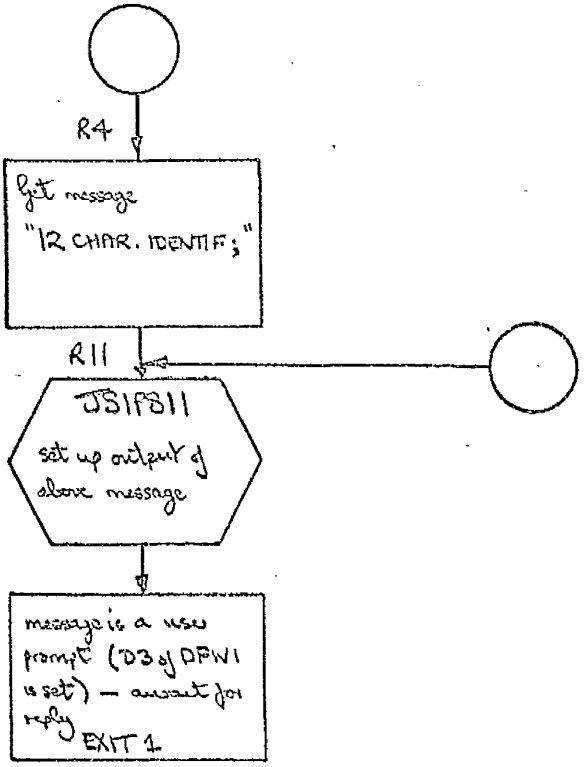

P840
JSP805
get time since midnight in tilittyps characters
Get current date and system identifier
Set up access to Node Jobs table in P801
JS6P801
allocate a fresh Temporary input file Work
Store FOC temporary input file in P801 table Prepare to set up message for new user
JS1P811
set up message
JS3
use R3 to mark terminal in receive status
JSP912
set up access to FIFO for this terminal
P73

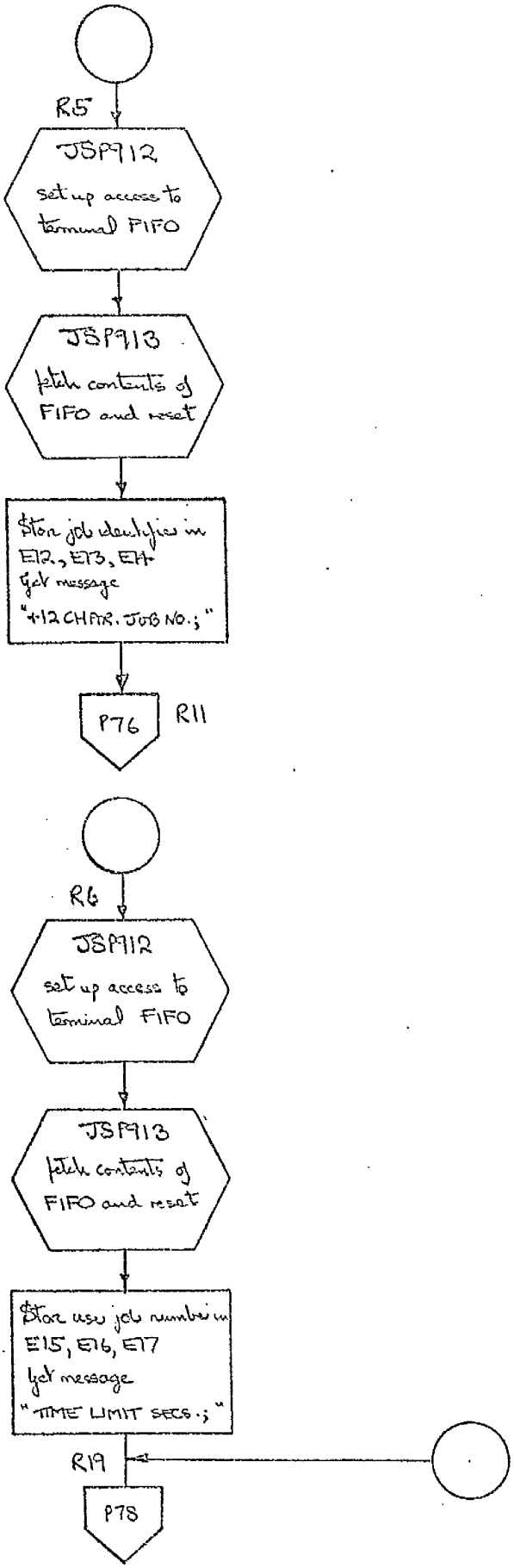

JSP913
reset FIFO to dummy characters
JSP806
access DFW1 and clear it
Get current terminal address in N1
JSP801
get address of FOC sector of temporary input file
JSP831
insert FOC (header) pointer for input file in I-part of DFW1
JSP807
set up DFW2 as ϕ|header|NOC
Store new DFW2
Reset SPCW to 3ϕ|1|ϕ in table
Put all ones in N1 as new value of GOAW
P74

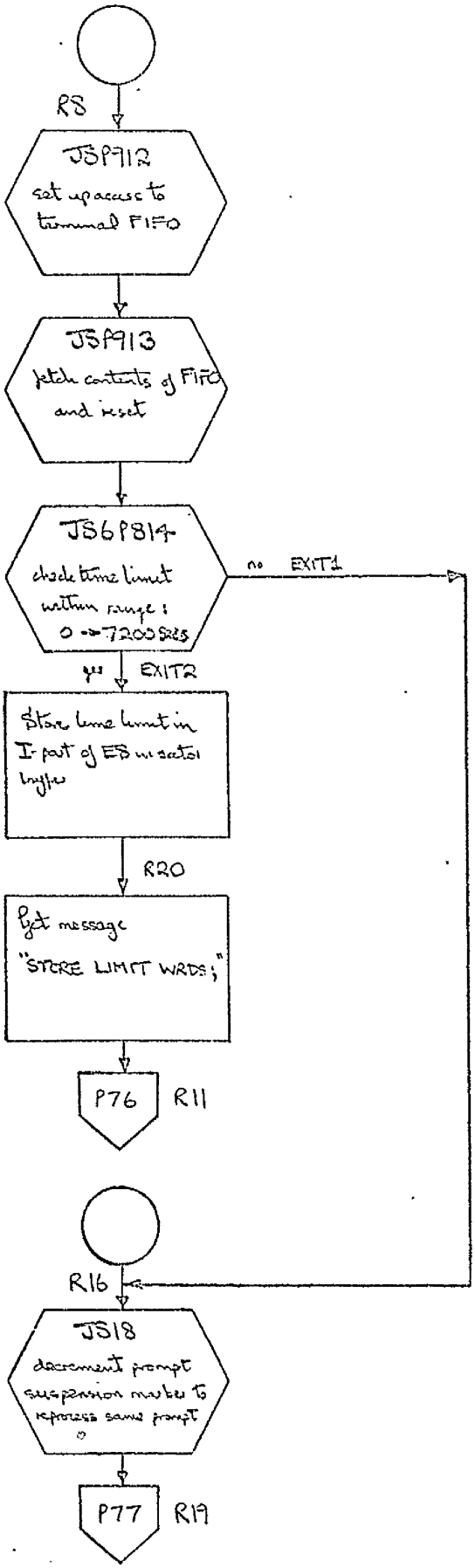

JS14P841
set GOPW = -1
$to terminal address in C7 and use C7 to set bit in V49P899.
V49P899 is a "logged-in" mask unused in the present version
EXIT 1
R3
Set current terminal address in C7
R4
Clear bit defined by C7 in N1 — leave result in N1
EXIT 1

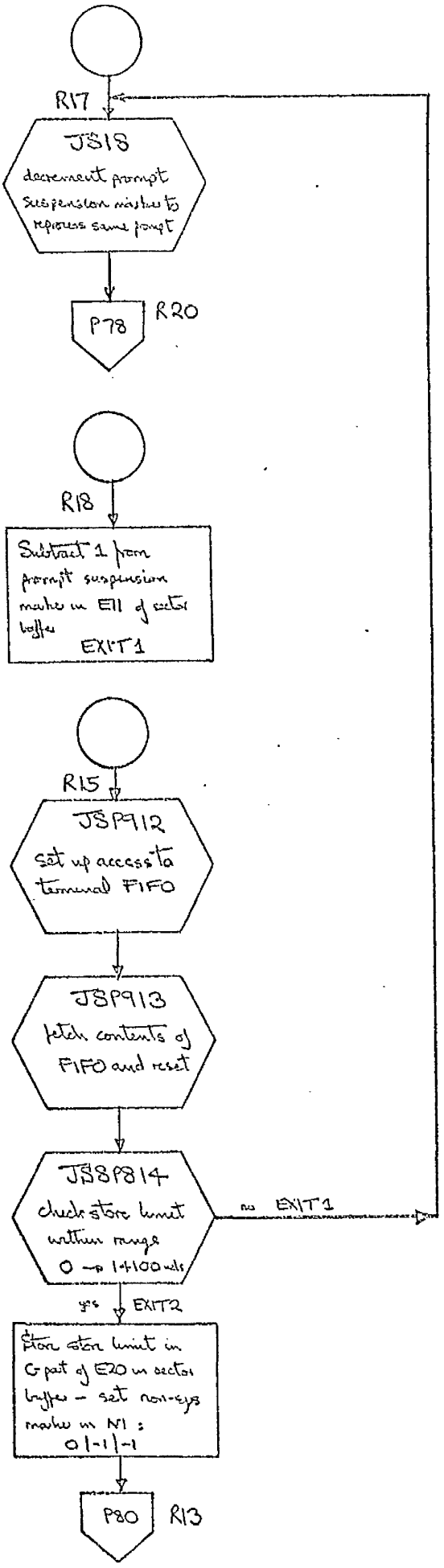

P84-1
JSP911
clear OCW 2 to zero
JSP806
get DFW1
Set D5-7 of DFW1 = ϕ ie command # ϕ
Set D3 = 1 - user response to prompt due
JSP906
set Q9 to sector buffer b/A pair this terminal
JSP910
check lockout on sector buffer defined by Q9
EXIT2 lockout
P910 does not return to point of call if lockout is set - see P910
no lockout
EXIT1
Clear prompt cycle reset inhibit marker in E22 of sector buffer and prompt suspension marker in E11 of sector buffer
P76
R21
R1
JSP906
set M9 to sector buffer b/a for this terminal
R21
P76

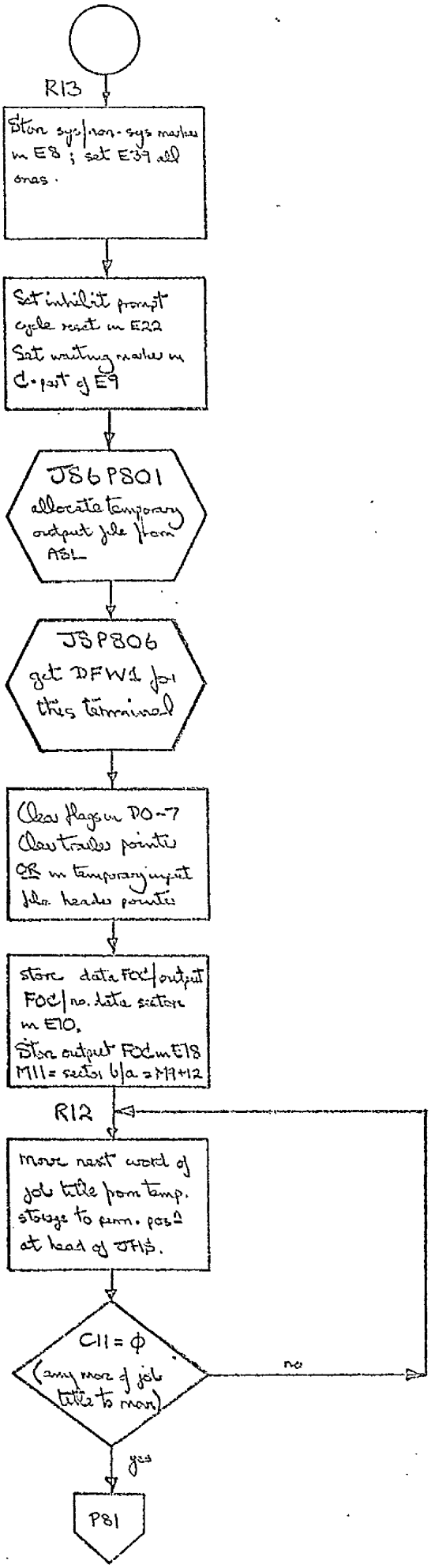

R21
Increment prompt suspension marker in E11 of sector buffer and store old value in Link
EXIT AR3
R3+0
R4
R3+1
P77
R5
R3+2
P77
R6
R3+3
P78
R8
R3++
P79
R15
R3+5
P82
R22

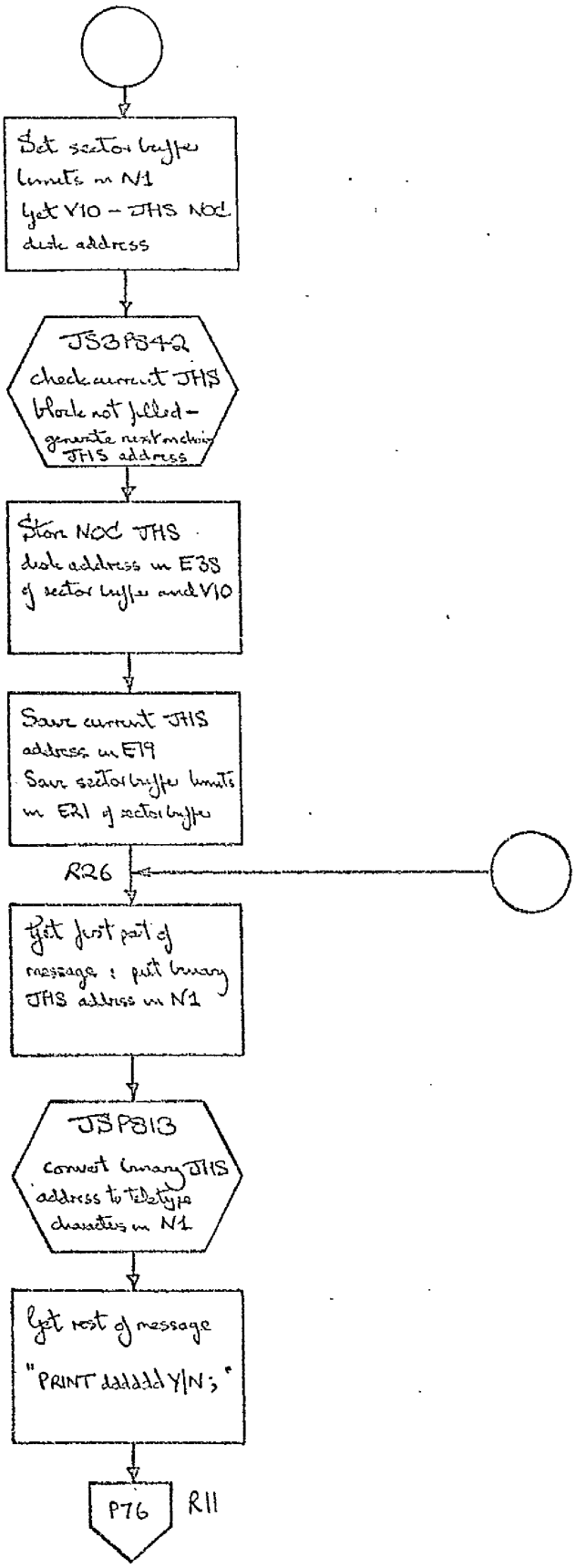

R4
Get message "12 CHAR. IDENTIF;"
R11
JSIFS11
set up output of above message
message is a user prompt (D3 of DFW1 is set) — await for reply
EXIT 1

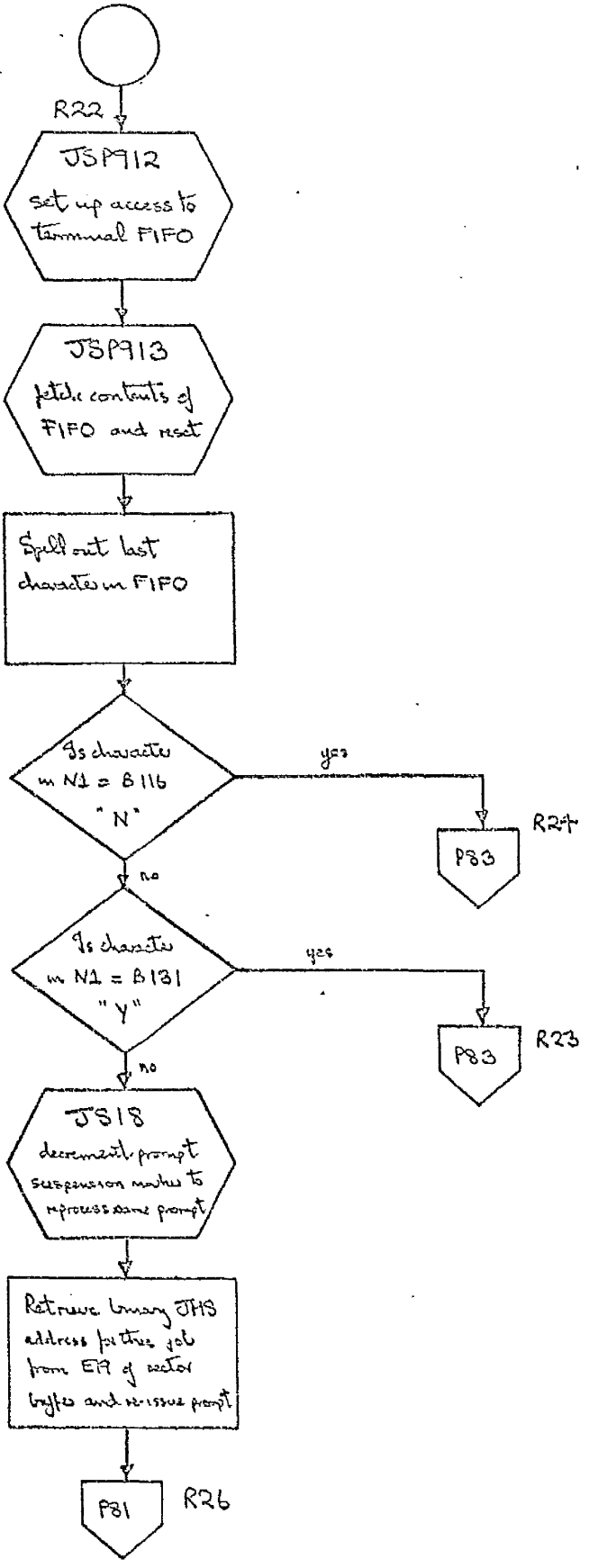

R5
JSP912
set up access to terminal FIFO
JSP913
fetch contents of FIFO and reset
$Stor job identifiers in E12, E13, E14
Get message
"+12 CHAR. JOB NO.;"
P76
R11
R6
JSP912
set up access to terminal FIFO
JSP913
fetch contents of FIFO and reset
$Stor user job number in E15, E16, E17
Get message
"TIME LIMIT SECS.;"
R19
P78

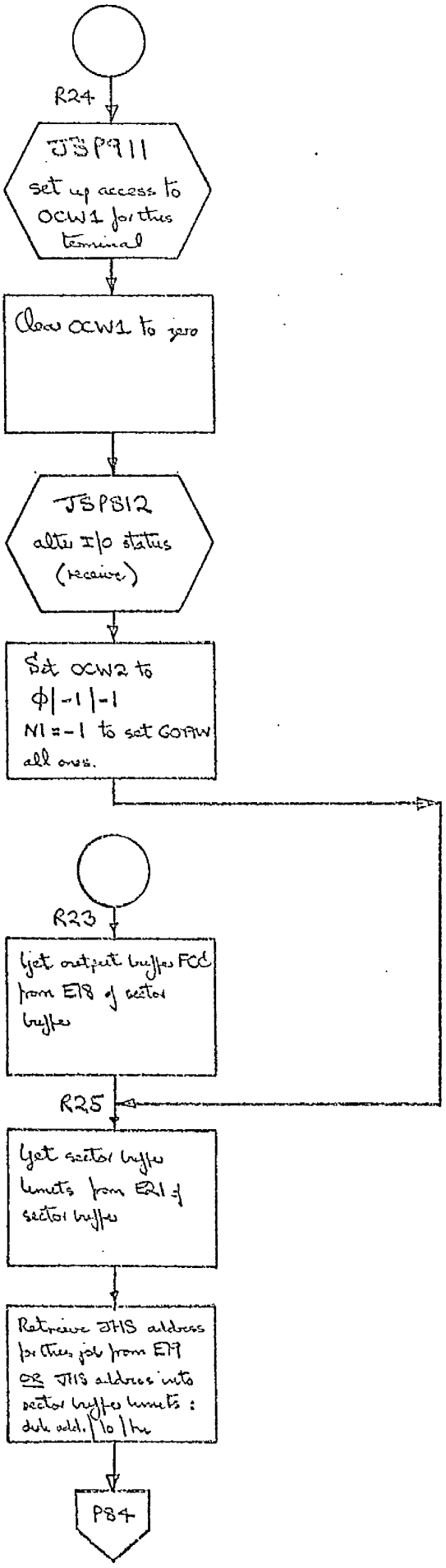

R8
JSP712
set up access to terminal FIFO
JSP713
fetch contents of FIFO and reset
JS6P814
check time limit within range: 0 -> 7200 secs
no
EXIT1
yes
EXIT2
store time limit in I-part of ES in sector buffer
R20
get message "STORE LIMIT WRDS:"
P76
R11
R16
JS18
decrement prompt suppression marker to reprocess same prompt
P77
R19

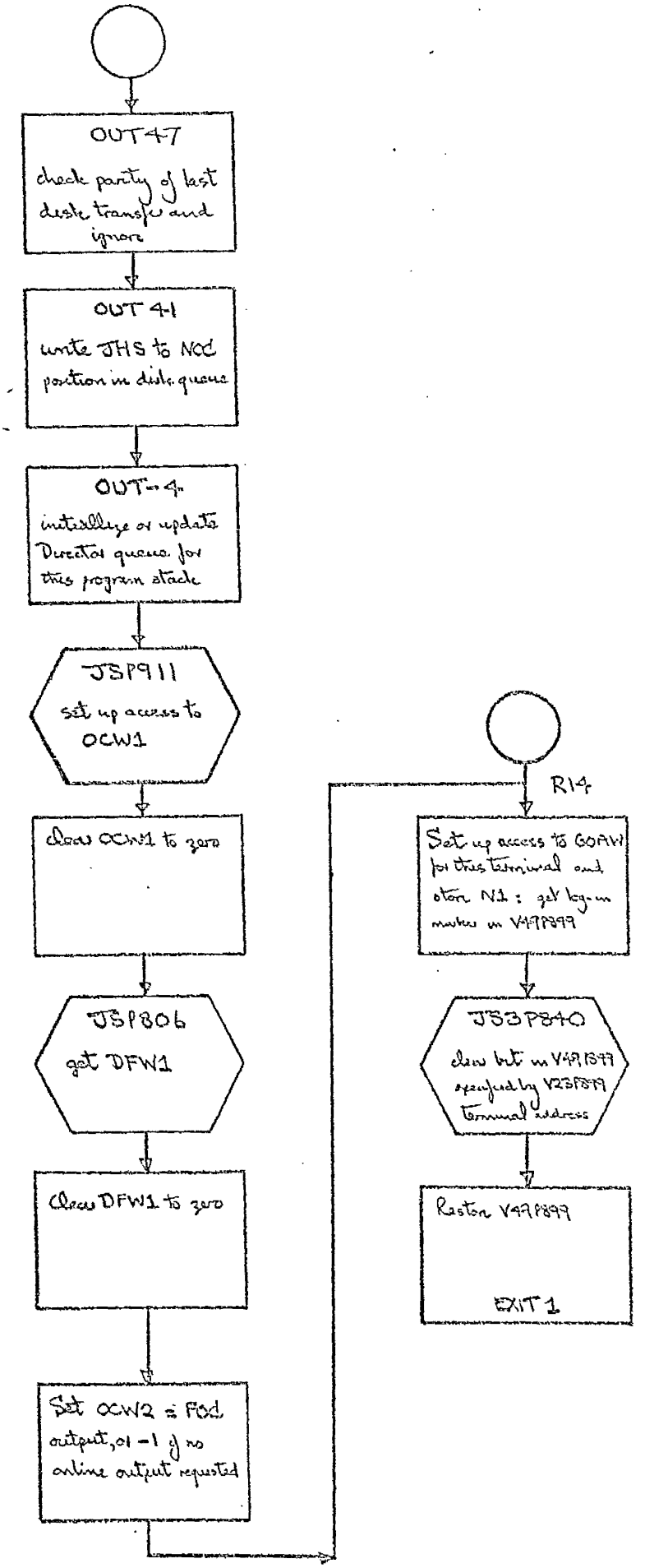

R17
JS18
decrement prompt suspension master to repress same prompt
P78
R20
R18
Subtract 1 from prompt suspension make in E11 of sector buffer
EXIT 1
R15
JSP912
set up access to terminal FIFO
JSP913
fetch contents of FIFO and reset
JSSP814
check store limit within range 0 → 1+100 wds
no
EXIT 1
yes
EXIT 2
Store store limit in G part of E20 in sector buffer – set non-sys make in N1 : 01-1|-1
P80
R13

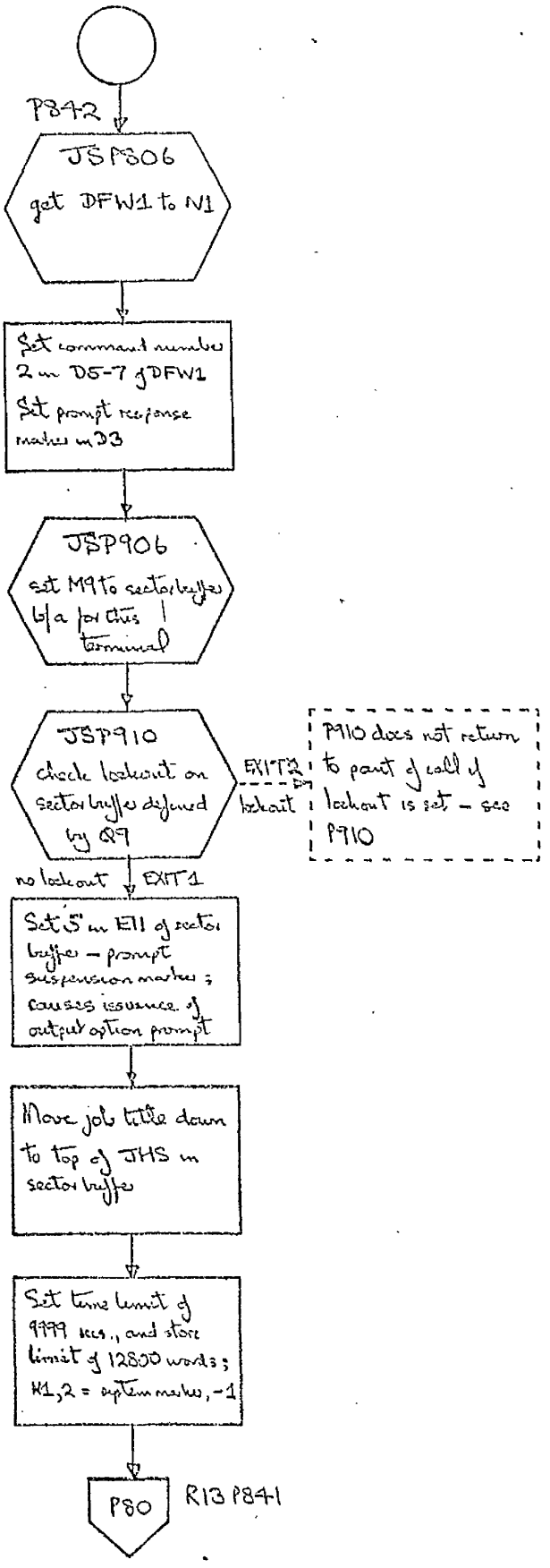

R13
Stor sys/non-sys marker in E8 ; set E39 all ones.
Set inhibit prompt cycle reset in E22
Set waiting marker in C-part of E9
JS6P801
allocate temporary output file from ASL
JSP806
get DFW1 for this terminal
Clear flags in P0-7
Clear trailer pointer
OR in temporary input file header pointer
store data FOC/output FOC/no. data sectors in E10.
Stor output FOC in E18
M11 = sector b/a = M9+12
R12
move next word of job title from temp. storage to term. pos^n at head of THP.
C11 = ϕ
(any more of job title to move)
no
yes
P81

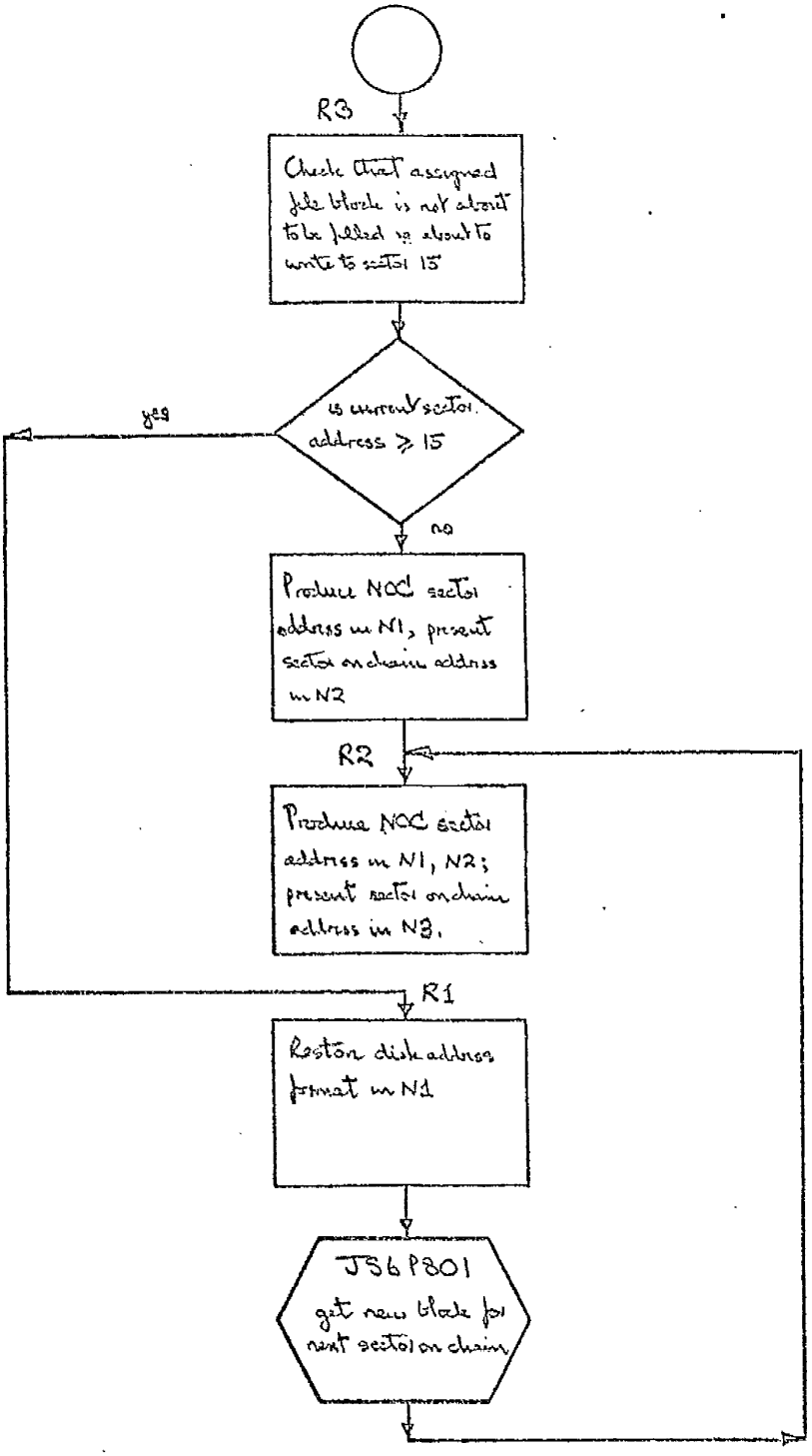

Set sector buffer limits in N1
Get VIO – JHS NOC disk address
JSP842
check current JHS block not filled – generate next/modify JHS address
Store NOC JHS disk address in E38 of sector buffer and VIO
Save current JHS address in E19
Save sector buffer limits in E21 of sector buffer
R26
Get first part of message ; put binary JHS address in N1
JSP813
convert binary JHS address to teletype characters in N1
Get rest of message "PRINT dddddd Y/N;"
P76
R11

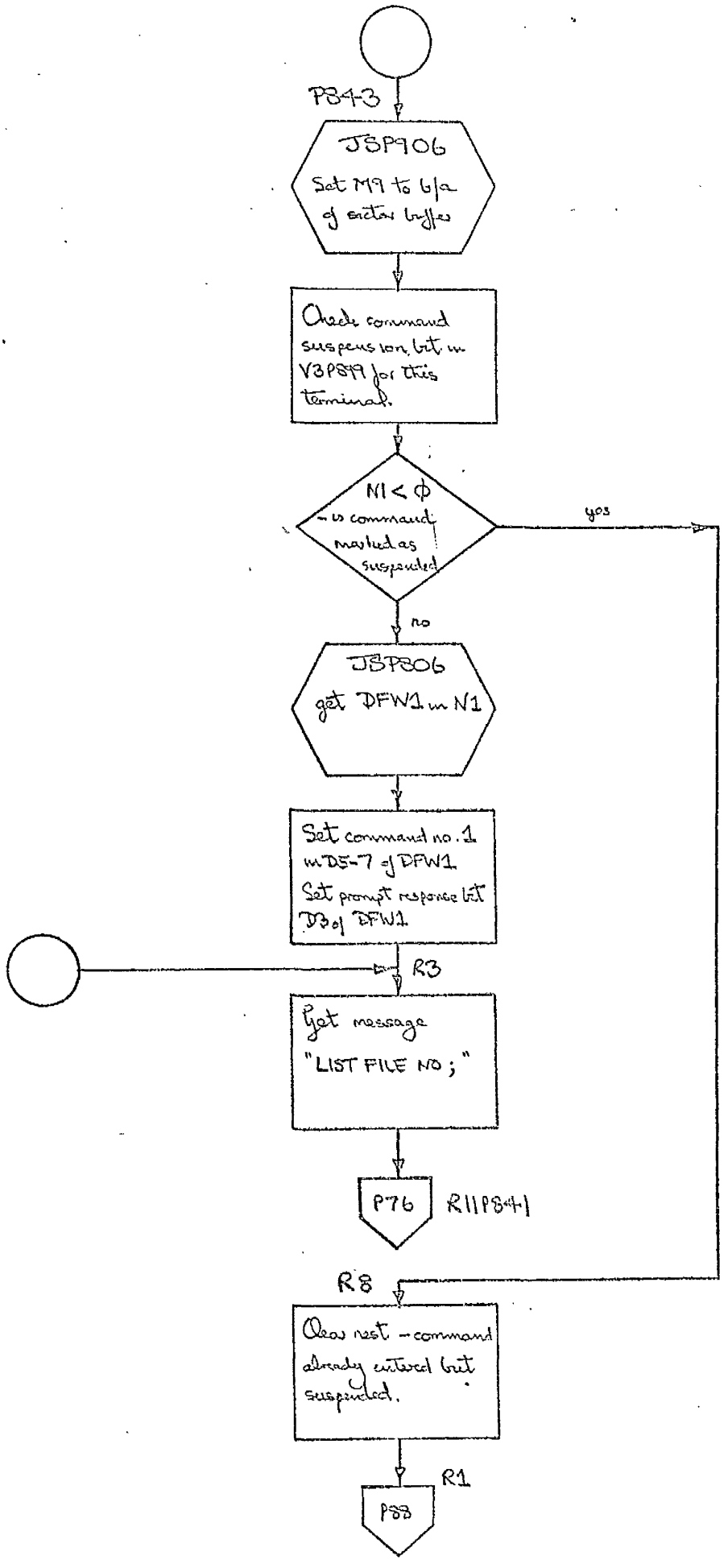

R22
JSP912
set up access to terminal FIFO
JSP913
fetch contents of FIFO and reset
Spill out last character in FIFO
Is character in N1 = B'116' "N"
yes
P83
R24
no
Is character in N1 = B'131' "Y"
yes
P83
R23
no
JS18
decrement prompt suspension marker to reprocess same prompt
Retrieve binary JHS address for this job from EA of vector byte and re-issue prompt
P81
R26

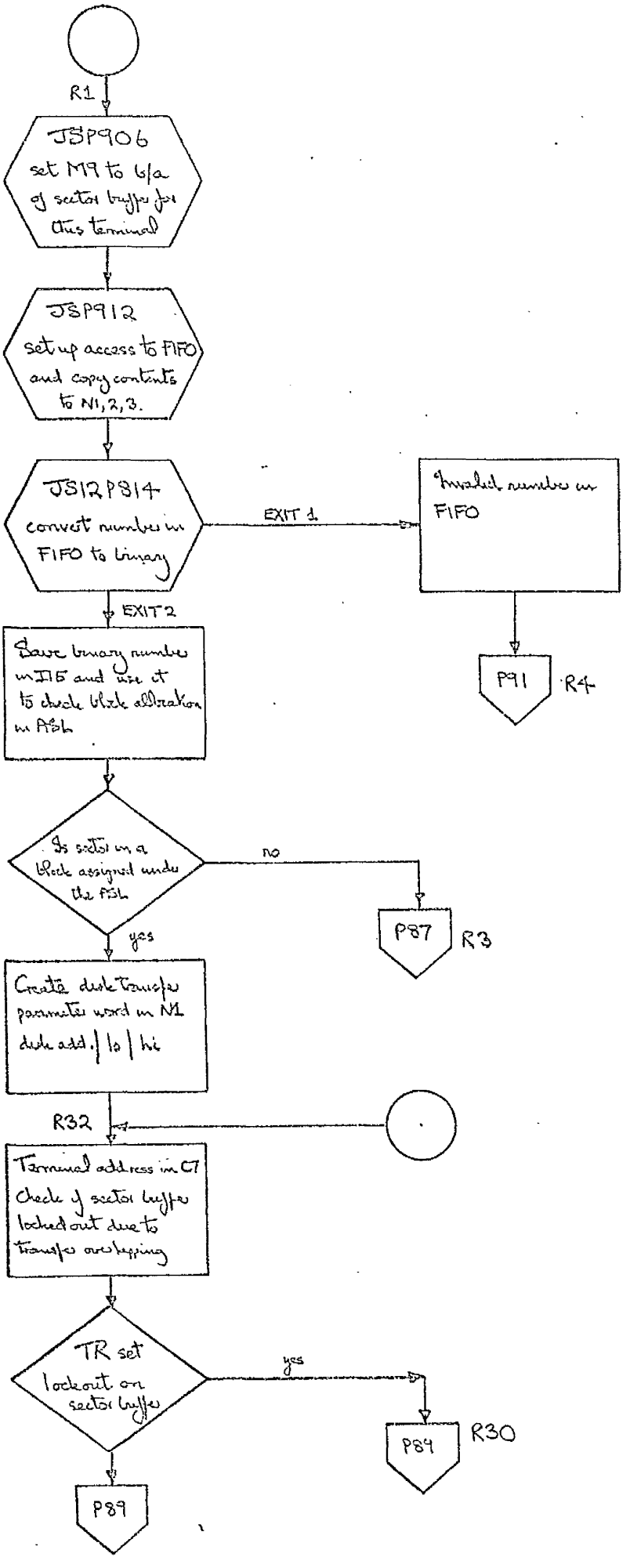

R24
JSP911
set up access to OCW1 for this terminal
Clear OCW1 to zero
JSPS12
alter I/O status (receive)
Set OCW2 to
ϕ|-1|-1
N1=-1 to set GOPW all ones.
R23
Get output buffer FCC from E18 of sector buffer
R25
Get sector buffer limits from E21 of sector buffer
Retrieve JHIS address for this job from E19
OR JHIS address into sector buffer limits:
disk addr. | lo | hi
P84

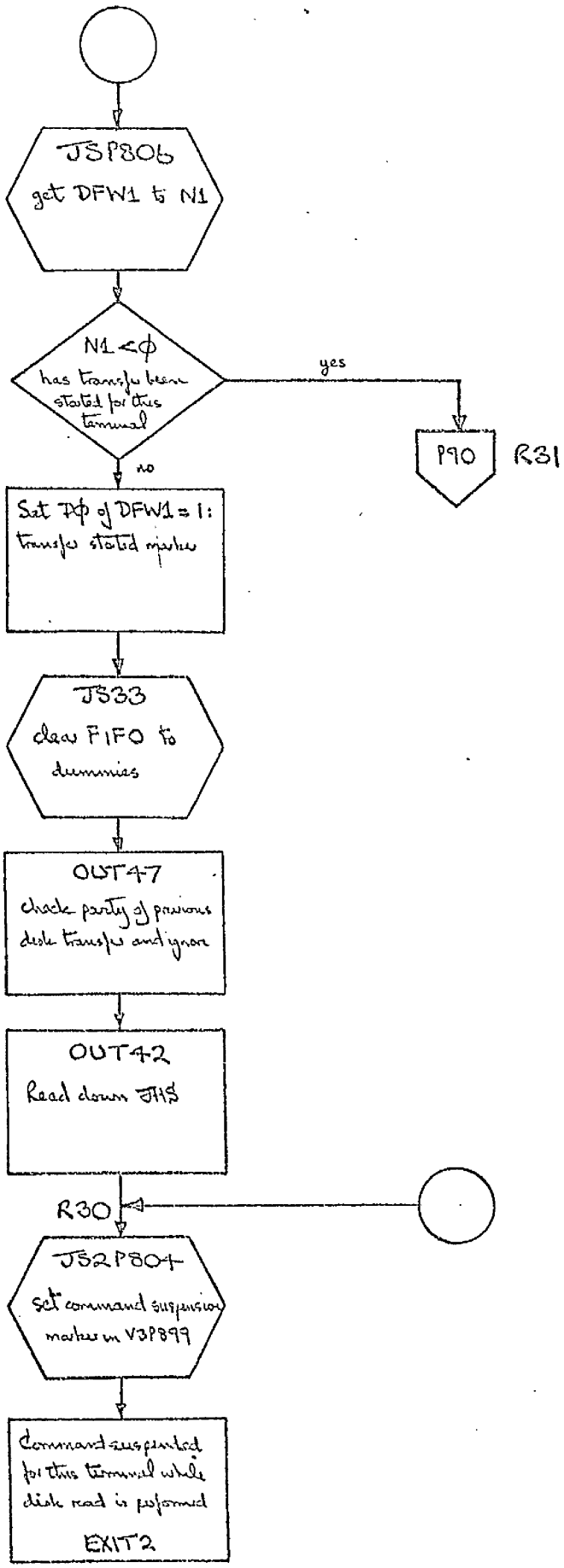

OUT47
check parity of last disk transfer and ignore
OUT41
write JHS to NOC position in disk queue
OUT-4-
initialize or update Director queue for this program stack
JSP911
set up access to OCW1
clear OCW1 to zero
JSP806
get DFW1
Clear DFW1 to zero
Set OCW2 = FOD output, or -1 if no online output requested
R14
Set up access to GOAW for this terminal and store N1 : get log-on number in V4P899
JS3P840
clear bit in V4P899 specified by V23P899 terminal address
Restore V4P899
EXIT 1

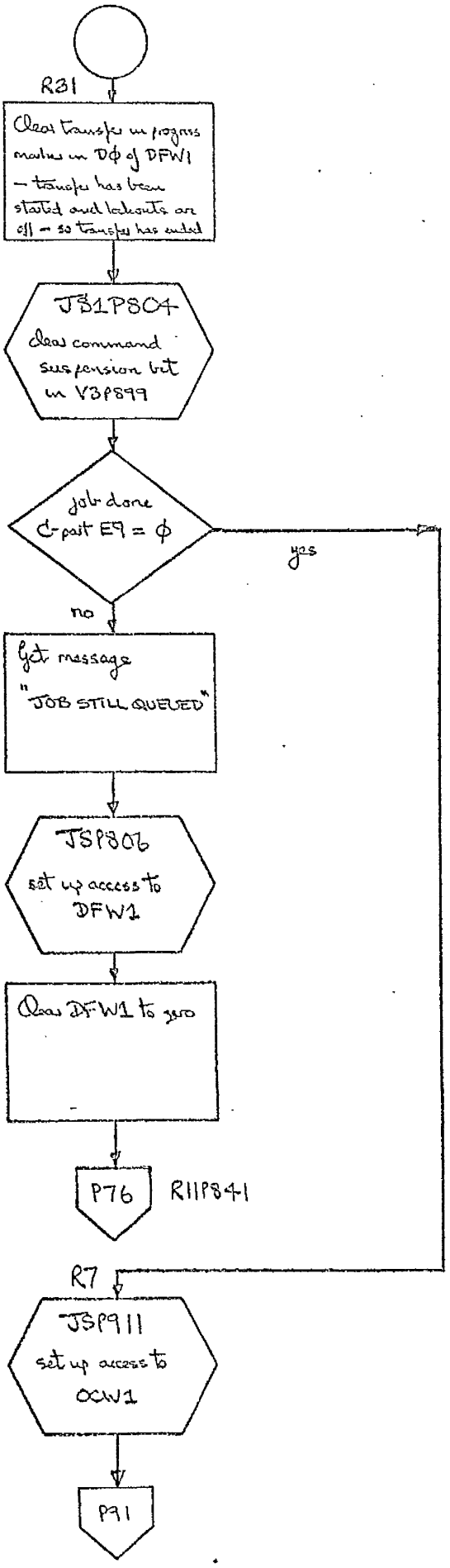

P842
JSP806
get DFW1 to N1
Set command number 2 in D5-7 of DFW1
Set prompt response marker in D3
JSP906
set M9 to sector buffer b/a for this terminal
JSP910
check lockout on sector buffer defined by Q9
EXIT2 lockout
P910 does not return to part of call if lockout is set – see P910
no lockout
EXIT1
Set '5' in E11 of sector buffer – prompt suspension marker; causes issuance of output option prompt
Move job title down to top of JHS in sector buffer
Set time limit of 9999 secs., and store limit of 12800 words; H1,2 = system marker, -1
P80
R13 P841

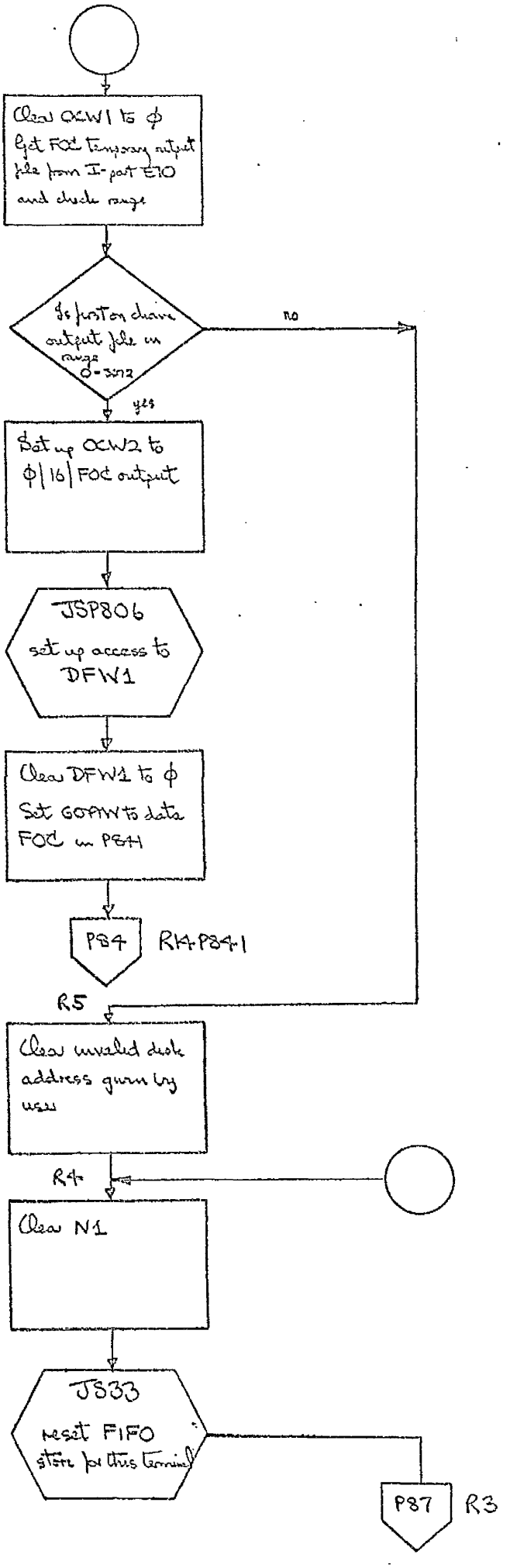

R3
Check that assigned file block is not about to be filled or about to write to sector 15
is current sector address ≥ 15
yes
no
Produce NOC sector address in N1, present sector on chain address in N2
R2
Produce NOC sector address in N1, N2; present sector on chain address in N3.
R1
Reston disk address format in N1
JSYS301
get new block for next sector on chain

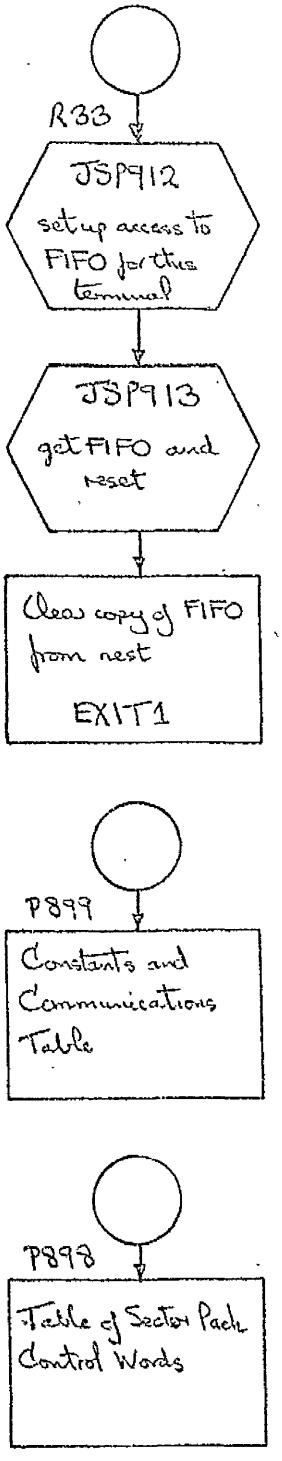

P843
JSP906
Set M9 to 6/a of sector bytes
Check command suspension bit in V3P899 for this terminal.
N1 < ϕ
- is command marked as suspended
yes
no
JSP806
get DFW1 in N1
Set command no. 1 in D5-7 of DFW1
Set prompt response bit D3 of DFW1
R3
Get message "LIST FILE NO ;"
P76
R11P841
R8
Clear rest - command already entered but suspended.
R1
P88

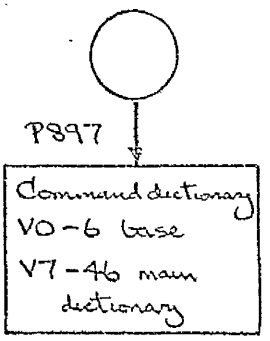

R1
JSP906
set M9 to b/a of sector buffer for this terminal
JSP912
set up access to FIFO and copy contents to N1,2,3.
JS12,P814
convert number in FIFO to binary
EXIT 1
Invalid number in FIFO
P91
R4
EXIT 2
Save binary number in N15 and use it to check block allocation in FAT
Is sector in a block assigned under the FAT
no
P87
R3
yes
Create disk transfer parameter word in N1
disk addr. | lo | hi
R32
Terminal address in C7
Check if sector buffer locked out due to transfer overlapping
TR set
lockout on sector buffer
yes
P84
R30
P89

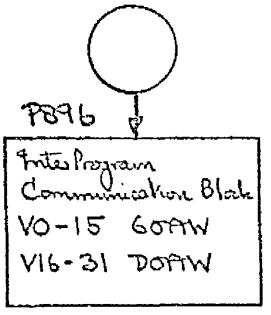

JSP806
get DFW1 to N1
N1<ϕ
has transfer been started for this terminal
yes
P90
R31
no
Set 7ϕ of DFW1 = 1:
transfer started marker
JS33
clear FIFO to dummies
OUT47
check parity of previous disk transfer and ignore
OUT42
Read down THS
R30
JS2P804
set command suspension marker in V3P899
command suspended for this terminal while disk read is performed
EXIT2


R31
Clear transfer in progress marker in DΦ of DFW1 – transfer has been started and lockouts are off – so transfer has ended
JS1PSC4
clear command suspension bit in V3P899
job done C-part E9 = Φ
yes
no
Get message "JOB STILL QUEUED"
JSP806
set up access to DFW1
Clear DFW1 to zero
P76
R11P841
R7
JSP911
set up access to OCW1
P91


Clear OCW1 to ϕ
Get FOC temporary output file from I-port E10 and check range
Is first or third output file in range 0-3ff2
no
yes
Set up OCW2 to ϕ|16|FOC output
JSP806
set up access to DFW1
Clear DFW1 to ϕ
Set GOPW to data FOC in P84
P84
R4+P84+1
R5
Clear invalid disk address given by user
R4+
Clear N1
JS33
reset FIFO status for this terminal
P87
R3


R33
JSP912
set up access to FIFO for the terminal
JSP913
get FIFO and reset
Clear copy of FIFO from rest
EXIT1
P899
Constants and Communications Table
P898
Table of Sector Pack Control Words


P897
Command dictionary
V0-6 base
V7-46 main
dictionary


P896
Fortes Program
Communication Block
V0-15 GOFTW
V16-31 DOFTW

APPENDIX II

Integrated Circuit Multiplexor, ICM

The use of a hardware multiplexor was a compromise between a wasteful "interrupt per character" system using the KDF9 alone and an expensive message handling computer which could not in any event have been installed before late 1967, over a year after we started work.

Each terminal is connected via a common half duplex data bus to the ICM in a passive manner and each must be actively interrogated in turn (polled) by the ICM to send or accept a character - there are no line contention problems and all asynchronous terminal operation is hidden by the ICM and the bus hardware.

The cyclically polled bus system is economical of hardware but precludes the attachment of devices of widely differing transmission rates unless special arrangements are made (e.g. CDC 3316 Communications Multiplexor Controller). The ICM could also operate in a burst mode of over 1000 chars./sec. when not in use by Common (see Chapter 4.4.2. P803).

Multicharacter ICM transfers and immediate teletype keyboard-printer feedback meant that echoplexing was impossible nor could any provision be made to blank-off any terminal on a temporary basis e.g. following receipt of a command terminator.

The length of transfer set up on the ICM governed the minimum response time of the system and was a compromise between short transfers with heavy overheads and long transfers with poorer response. There was the advantage that transfer lengths and therefore times could be dynamically altered under program control to reflect the amount of online activity, increasing as processing demands in CommCon increased.

The ICM operated by fetching 12 bit fields from the core output buffer and inspecting the character field (see Fig. 15.): if the character was 377_8 then this was interpreted as a read request from the terminal addressed in the address field. When the user presses a teletype key the bit pattern corresponding to the character is assembled in shift register 1 and moved to shift register 2 where it remains until interrogation control for this terminal recognizes its address on the data bus; the character is then gated onto the data bus and across to the write register which causes it to be printed. The ICM picks off the incoming character from the data bus along with its terminal address and deposits this 12 bit field in the core input buffer. If the output character was not a 377_8 then it is transmitted on the data bus with its terminal address and the correct interrogation control picks it off and

and gates it to the terminal write register causing it to be printed.

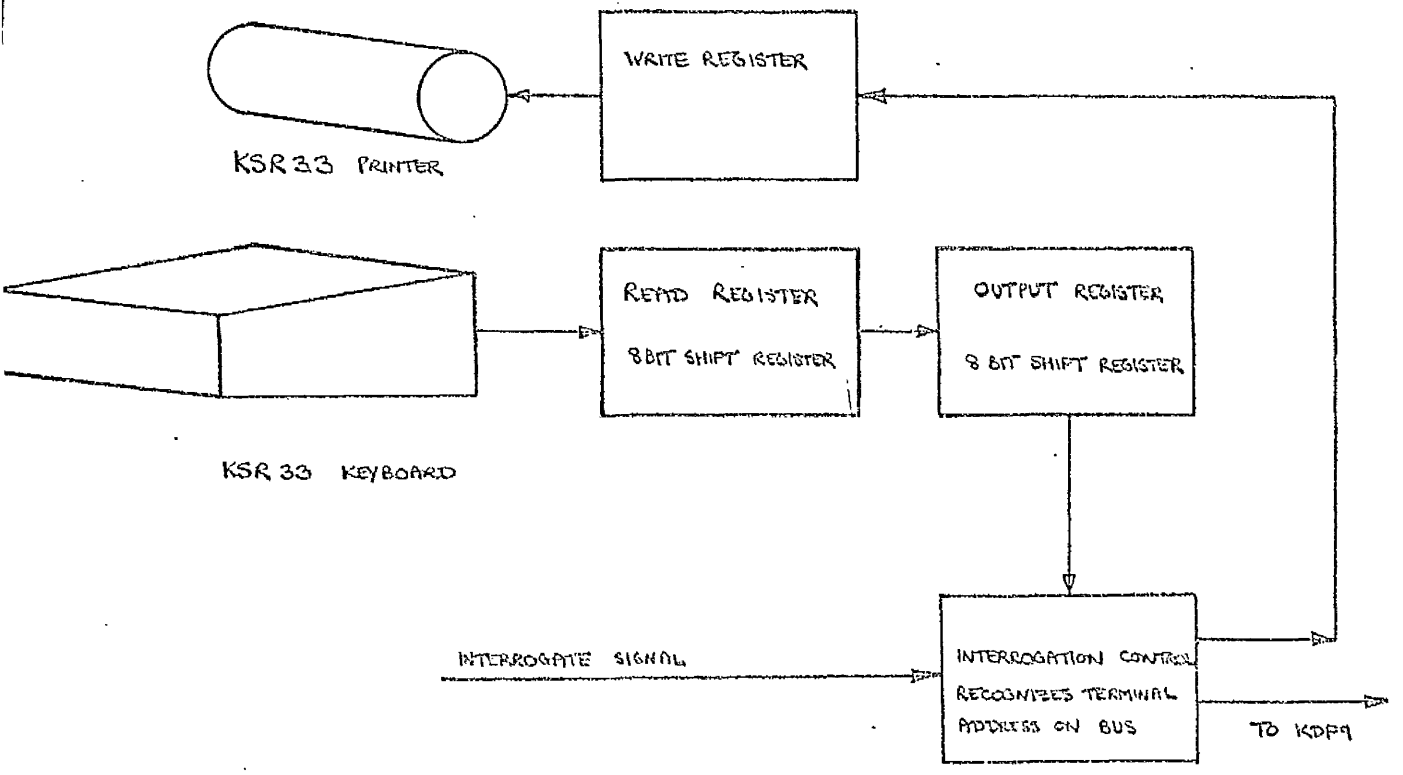


Fig. 15. Logic of the ICM

Addresses 00_8 to 16_8 were scheduled to take KSR 33 teletypes; address 17_8 was reserved for use as a change-mode instruction to allow burst-mode operation: this feature was not used in Phase One.

The ICM was designed and built within the Department using integrated circuits and wire wrap construction; it was connected to channels 2 and 4 of the KDF9 I/O control (normally PTR and CP) and was handled like an ordinary peripheral by the user program

using paper tape read/write instructions of the form PWQq; PRQq; in that order.

The main advantage of the ICM were its low cost, easy implementation and immediate availability; the main disadvantages were that it threw too heavy a load on the KDF9 to be efficient and flexible and that there was no provision to blank selected terminals on a temporary basis.

APPENDIX III

The following subsections give brief details of the components of the operating system developed by Mr. Mathovits. With the exception of the segmented Director all components run in program mode and have privileged capabilities, e.g. access to Director constants, interprogram disk access etc.

The segment loader, SEGLOAD06UPU

The first program to enter the machine after resident Director has been loaded must be the segment loader; any TINT performed before this forces entry to P52, the TINT T subroutine which calls for a B-type program. The operator must then load the segment loader. The segment loader claims three disks and loads up a sequence of 60 word binary segments starting at logical address 0; it then performs an OUT-7 to notify Director of its presence and Director notes the corresponding stack letter (this must be "S"). The loader then performs an OUT+1 and calls in the pseudo-offline loader, OLOADSO------.

POL loader, OLOADSO--UPU

This program is permanently resident in core with program letter "S". Its function is to spool jobs into the input well on the disks claimed by the

segment loader. Each job consists of a standard A-block as a call tape, followed by any number of variable length data blocks each delimited by one end-message character. Between each job is an eight character marker block of the form:

$$0505050505050505_8$$

The first block after a marker is read into a separate A-block buffer and the subsequent data is read thirty words at a time into another buffer and stored in the input well as chained 40 word sectors of the format shown in Fig. 16

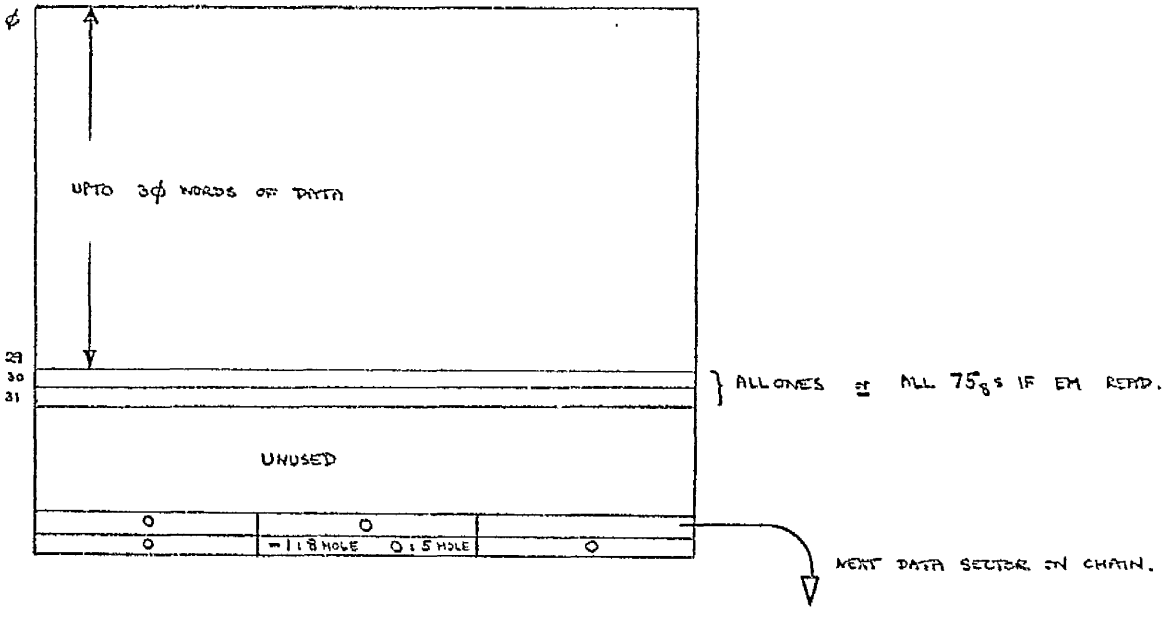


Fig. 16. Layout of input well data sector.

When a marker block is detected, the segment loader saves its own coding to disk and performs an OUT+1 to load and enter OLOADE1----- leaving the original constants and A-block buffer of OLOADEO----- unaltered in

core. OLOADER0------ also regularly inspects E/L for a TINT B.

TINT B1 releases the POL paper tape reader and terminates POL loading

TINT B2 claims a paper tape reader and resumes POL loading

TINT B3 claims a type 65 magnetic tape, relabels it <ZERO> and deallocates it.

POL loader overlay, OLOADER1---UPU

This overlay is loaded by OLOADER0---UPU. If the input medium letter of the program defined in the A-block buffer is "M" then the overlay types the query "PROGTAPE FOR <identifier> ;" on the monitor flexowriter and expects an eight character tape label in reply. The confirmatory query "/<identifier> CHECK;" is typed and the reply should be "Y" or "N". OLOADER1---UPU then creates a Job Loading Sector and writes it to disk in the next job queue position as defined by the constants of OLOADER0---UPU and performs an OUT-4 to initiate or update the job queue for its own program stack. Finally the coding of OLOADER0---UPU is reloaded from disk and re-entered.

Director, CHOICE---UPU

This is an expanded version of KERNEL---UPU with additional facilities listed below. Further details may

be found in Refs. 1, 2, 3, and the annotated coding of SEGMDIR--UPU produced by me.

SEGMDIR--UPU involves seven major areas of change from the standard Time Sharing Disk Director, KITSOX----UPU.

(i) introduction of time-sliced priority swapping between program stacks 2 and 8 as part of the round robin program running scheme.

(ii) provision of 12 segments and one 80 word segment area.

(iii) provision of trapped paper-tape-reader operations from the input well on disk.

(iv) provision of an extra set of OUTs with negative identifiers.

(v) provision of segment control coding.

(vi) new OUT+8 facilities with negative stream numbers allowing online output of terminal directed output.

(vii) modifications to allow the scheduler, JOBONG----UPU, to nominate a program for running from the job queues. Operator initiated jobs were run before RJE system jobs which were in turn run before POL loaded jobs.

Each segment contained one or more pure code procedures and all were designed to run to completion. With the exception of TINT 4, all TINT routines were segmented out and the coding of P4 altered to reject a

second TINT while processing of the first was still active i.e. any preceeding TINT segment required must have been loaded, entered and completed before a subsequent TINT could be accepted. This avoided queuing problems in Director subprogram 0 (the TINT subprogram).

OUT 0, 1, 2, 6, 7 are segmented out and the issuing progrom is suspended in its main Director subprogram using the standard facilities of P10 and P11 awaiting clearance of lockout on a "segment-active" marker in P106.

While any segment is being loaded or is active lockout is set on the first word of P106: segment 3, TINT 0, further sets up a transfer within the segment area before clearing lockout on P106 marking the segment as completed. The correct sequence for P11 is, therefore, to test lockout on P106, test lockout on the segment area for intra-segment transfers and finally test lockout on P106 again. The second test on P106 is required because under control of P11 a high priority subprogram may call for segment loading after the TINT 0 segment ends but before control again passes to a low priority subprogram which is suspended on lockouts within the segment area.

A full list of segments and functions of the negative OUTs is shown in Table 1 and Table 2.

In Phase One, I/O for spooled jobs was handled by Director via the temporary file system in the following: input was limited by JOB Organizer to a maximum of sixty four disk sectors and output to sixteen disk sectors by Director for negative OUT=0 streams; there was no limit on normal OUTs.

TABLE 1.

SEGMENT NO:	CONTENTS
1	TINTS BEJRS
2	TINTS GIKLP
3	TINTS UV
4	OUTS 0,2
5	P9, P12
6	TINT A, OUT 0 (modified)
7	OUT 1, 6, 7
8	terminal directed OUT+8
9	director contents (used by JOBORO)
10	time sliding segment (stacks R and Q at 10 sec. intervals).
11	OUT-1 segment (program self suspension, nominate prog. tape)
12	TINTS CDMEX set No. of levels for queued jobs TINT M punch current segment

Director was generated by compiling twelve separate versions each containing one of the above segments: each version was then loaded under initial input and an immediate TINT M performed to punch out the relevant segment on the PTP.

All segments were collected in order and preceded by a 2 digit octal number terminated by an "→" which was the total number of segments to be loaded: in this case, 14 →, the format required by [illegible]LOAD06UPU.

TABLE 2.

OUT Number	
-1	N2=0 allows a program to suspend itself (D36 of V25P104N5)
	N2=1 allows a program to nominate a magnetic tape as a program input device
-2	unused
-3	N2=-1 get input well job queues constants from Director
	N2= 0 update input well job queues constants in Director
-4	N2= JHS address : initialliso or increment a job queue
-5	N2=-1 disk write to disks of program stack specified in N3
	N2= 0 disk read from disks of program stack specified in N3
-6	see listing : defino next job to be run in Director P120
-7	identify program stack of OLOADBO--UPU
-8	unused
-9	N2= IPCB Base address - identify CommCon
-10	set system marker

Job Organiser, JOBORG---UPU

JOBORG's task is basically that of scheduling: to select the next job to be run from one of the four queues held in V0 to V3 of Director subroutine F50%. The four queue pointers and the total number of entries are obtained by OUT-3 and JOBORG scans the queues in the order P, Q, R and S i.e. since OpenCon runs in P and OLOADER---UPU in S, online jobs are run in preference to offline jobs.

The highest priority queue with at least one entry is found and OUT-5 used to read down a block of upto 16 Job Reading Sectors from that queue on the disks. This queue of JRS is scanned for the first unstarted job and details of this job assembled in the next. If necessary, the JRS is code converted to Flexowriter code as is the associated implicit data file. Code conversion is done by JOBORG in preference to anywhere else because all jobs to be run (except operator jobs) must pass through JOBORG whether RWE jobs or POL jobs. Having selected the next job, JOBORG uses OUT-3 to update Directors job queue statistics, marks the chosen JRS as "done" and writes it back to disk, and finally performs OUT-6 to store details of the chosen job into V17-V25 of Director's P120 to initiate running of that job.

Work on devloping a scheduling algorithm and parasitic dumping progress to allow JOSCAR to control the round-robin time-slicing program running scheme was in progress when the project ended.

APPENDIX IV

The Backward Scanning Editor.

Full details of this package are not included here as basic coding was only just completed when the project terminated and was never tested.

The editor was intended to exploit the temporary file structure of backward (newest to oldest, lexicographically) pointing chain links to yield an interpretive editing language of a more natural structure than that available under the PROMPT answerer, by allowing the user to refer to lines of text by scanning backwards from the end of the file. Thus the user was free to insert file editing commands anywhere in the body of the file as he typed the file in: each correction message referred to a preceeding line and that line could equally be a further correction message acting yet further back in the file or a simple line of text. Line editing and character editing facilities were provided.

The editor read the file tail first from the last block on chain setting up a line-by-line list structure in core, the list was scanned tail first line-by-line until a command was found and interpreted. Line insertion and deletion was done by standard list processing techniques.

A command which referred to a section of the file not yet read down into core was re-chained onto a special list head. When the list currently in core was completely scanned it was repacked into standard file format, written to disk and the next sectors-on-chain in the backward direction read down. The list generation process was reported, the outstanding commands in the special list head chained onto the new tail and command searching resumed. Provided all commands had been successfully obeyed the old file was deleted and the new one saved. Failing commands could themselves be corrected by the same procedure.

The possibility of incorporating a macrogenerator into the editor was envisaged.

APPENDIX V.

BIBLIOGRAPHY

There can be no better bibliography than that compiled by George R. Trimble, Jnr., and published in COMPUTING REVIEWS Vol. 9, No. 5, May 1968, p. 291. In addition the following are recommended

M.V. Wilkes Time Sharing Computer Systems
MacDonald Press

ACM Operating System Symposium
Comm.ACM, 11, 5 (May 1968), 295-377.

REFERENCES

(1) EELM KDF9 Service Routine Library Manual
Vol. 2, Section 10.

(2) EELM KDF9 Non Time Shared Director Support
Documentation

(3) EELM KDF9 Time Shared Director Support
Documentation

(4) W.S. Bovie Batch Processing for Teletype ALGOL
M.Sc. Thesis Glasgow University 1967.